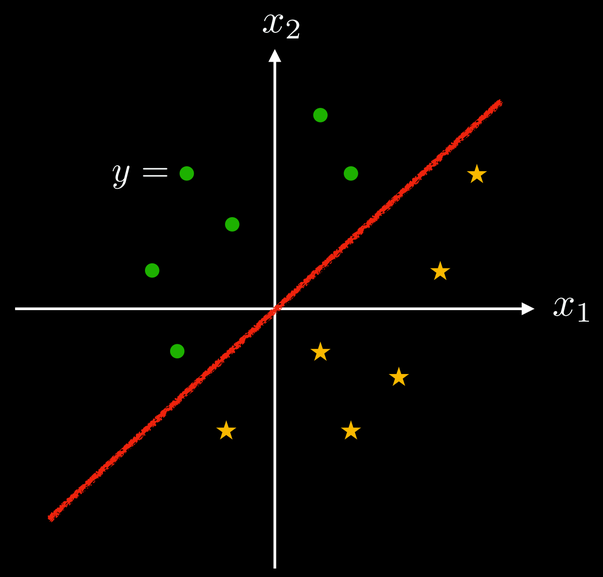
Deep Learning

Supervised Learning

Given $(\mathbf{x}, \mathbf{y})$

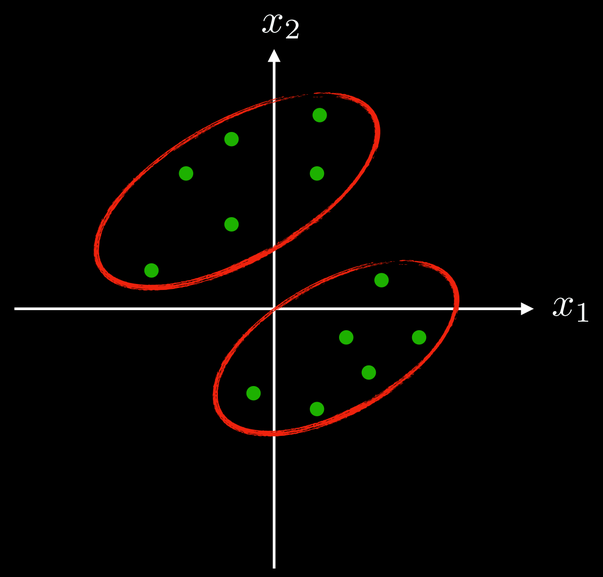


Regression

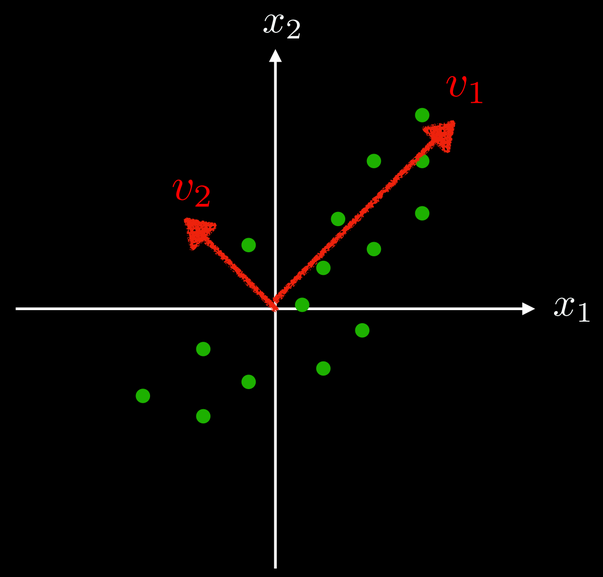
- Learn mapping given the labels
- Use to predict unseen labels

Unsupervised Learning

Given $\mathbf{x}$



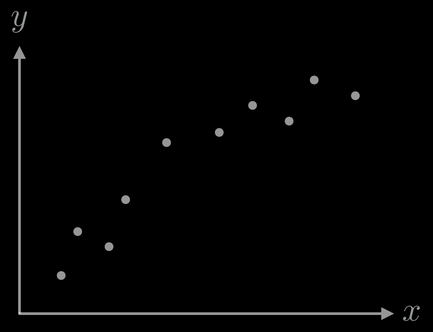
Clustering



PCA

- Find patterns in high dimensional data
- Use for dimensionality reduction

The ML workflow



Can we automate feature extraction?

$$\phi(x) = [1, x, \ldots, ?]$$

not satisfied

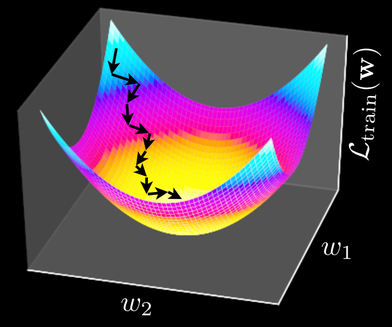
η

T λ

$\mathcal{L}_{\text{val}}$

satisfied

$\mathcal{L}_{\text{test}}$



Linear Predictor

$$f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)$$

$$\phi(x) = [1,\ x]$$

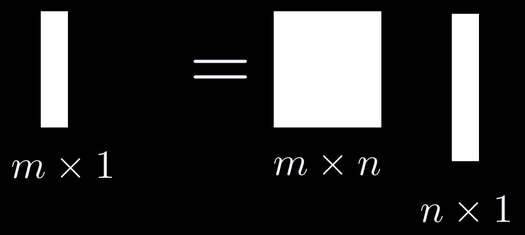
$$\phi(x) = [1, x, x^2, x^3]$$

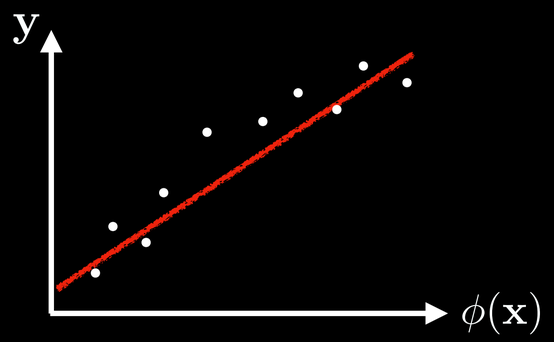
$$\phi(x) = [1,\ x,\ \sin(3x)]$$

Higher dimensions

$$\hat{\mathbf{y}} \qquad \mathbf{x}$$

$$f_{\mathbf{W}}(\mathbf{x}) = \mathbf{W}\ \phi(\mathbf{x})$$





Linear Predictor

Matrix multiplication

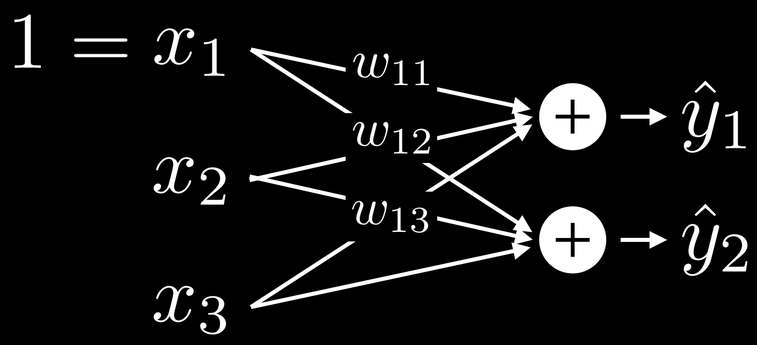
$$\hat{\mathbf{y}} = \mathbf{W}\mathbf{x}$$

Index notation

$$\hat{y}_i = \sum_{j=1}^{n} w_{ij} x_j$$

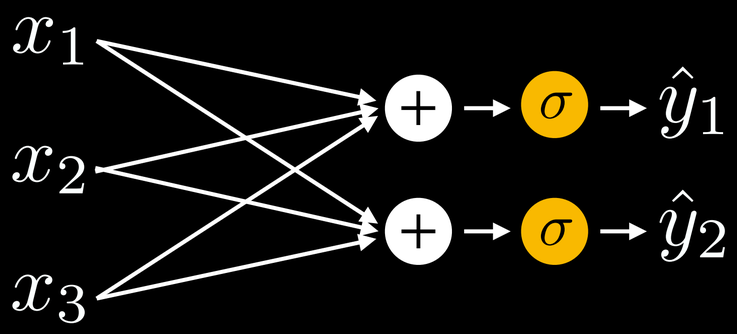
Define features such that
first component accounts for bias

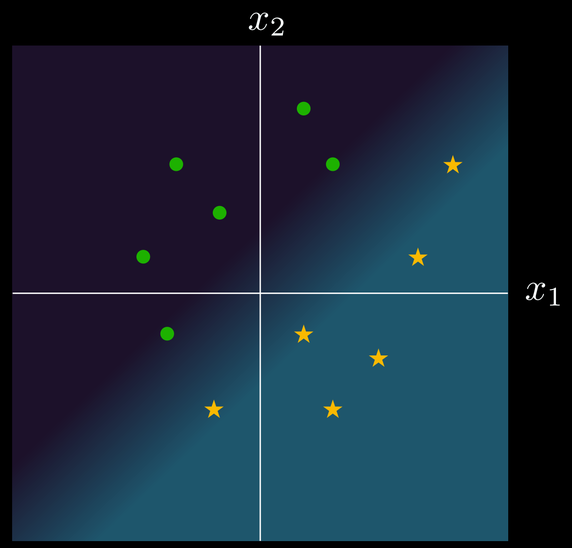
Network representation



Nonlinear Predictor

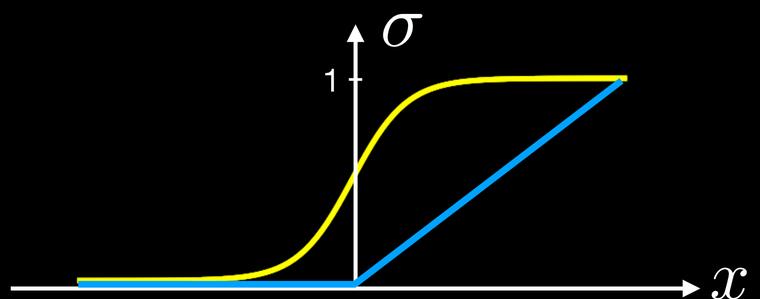
$$\hat{\mathbf{y}} = \sigma(\mathbf{W}\mathbf{x})$$





Logistic function: $\sigma(x) = \dfrac{1}{1+e^{-x}}$

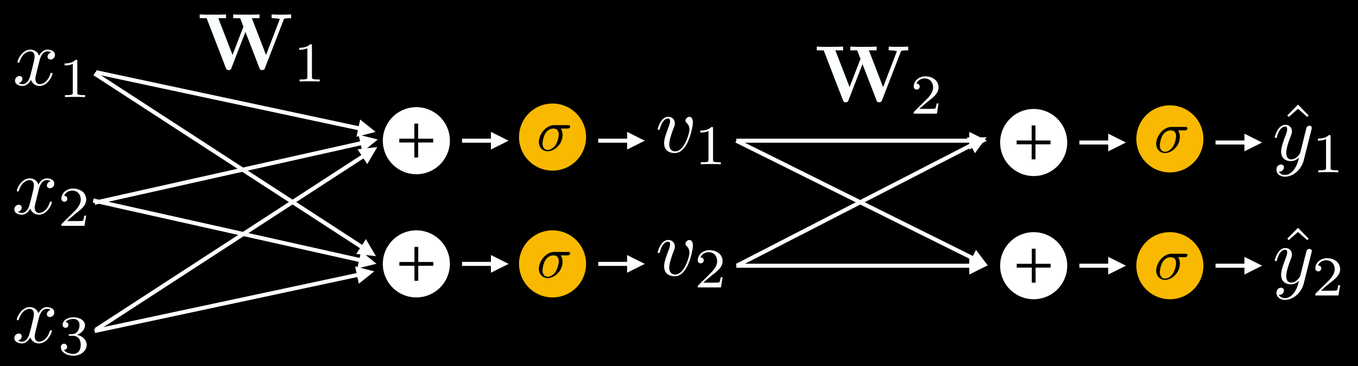
ReLU: $\sigma(x) = xH(x)$



Activation Function

Neural network

$$\hat{\mathbf{y}} = \sigma\left(\mathbf{W}_2\ \sigma\left(\mathbf{W}_1\mathbf{x}\right)\right)$$



$$\mathbf{x} \rightarrow \boxed{\sigma \circ \mathbf{W}_2 \circ \sigma \circ \mathbf{W}_1} \rightarrow \hat{\mathbf{y}}$$

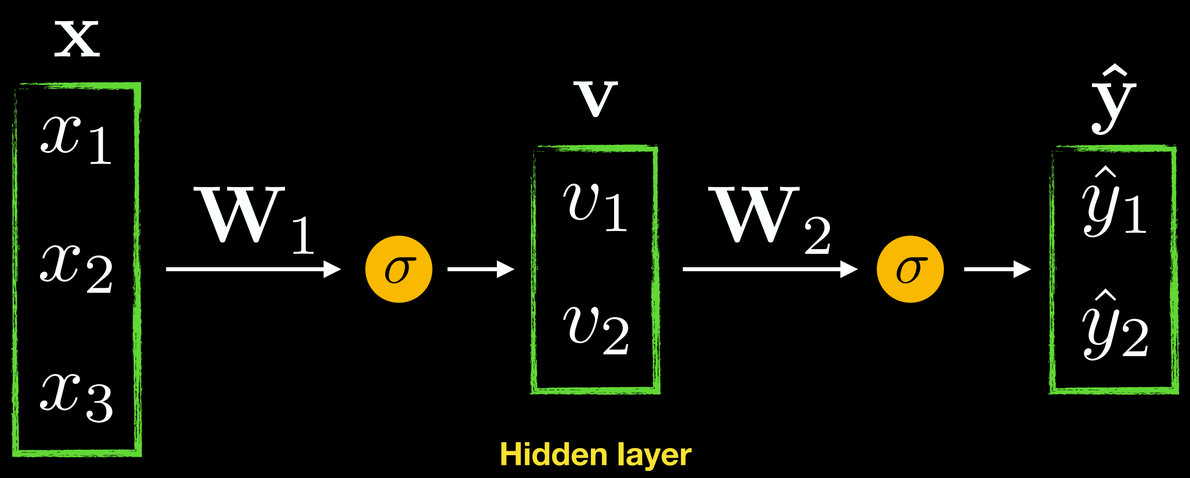
Nonlinear Linear Nonlinear Linear

Neural network

$$\hat{\mathbf{y}} = \sigma\left(\mathbf{W}_2\ \sigma\left(\mathbf{W}_1\mathbf{x}\right)\right)$$

$$\mathbf{v} = \sigma(\mathbf{W}_1\mathbf{x})$$

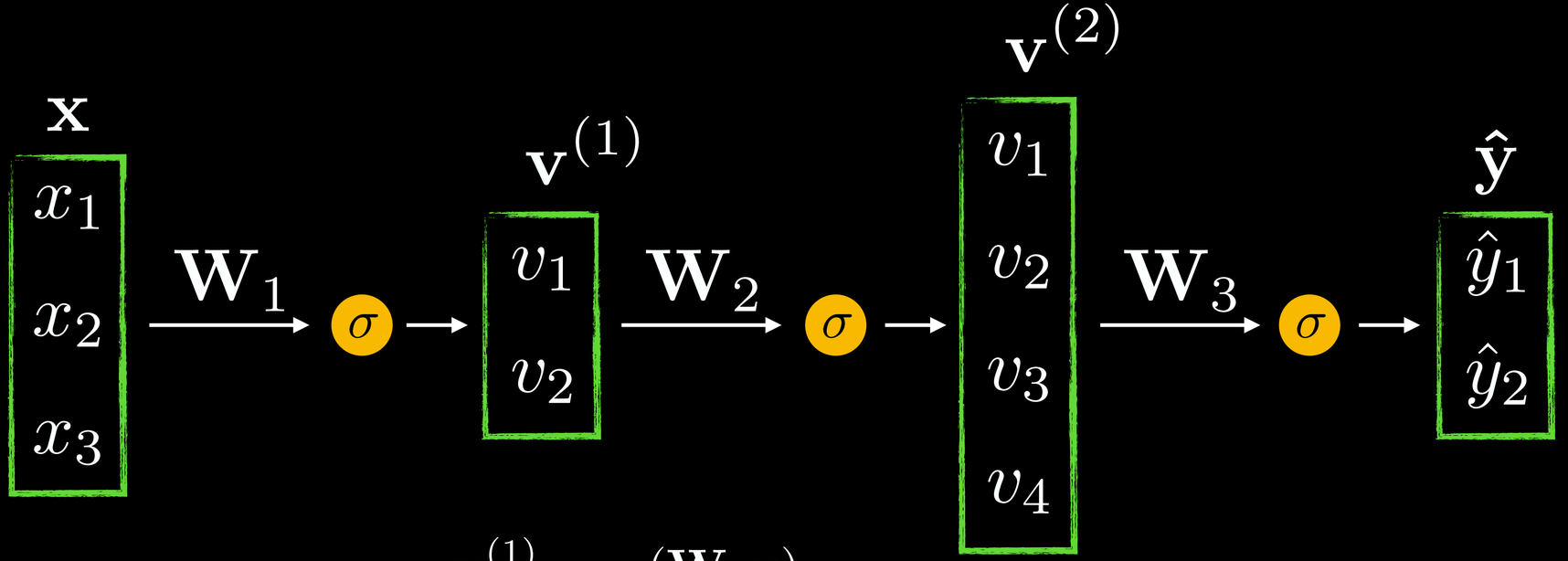
$$\hat{\mathbf{y}} = \sigma(\mathbf{W}_2\mathbf{v})$$



Can be interpreted as a learned $\phi(\mathbf{x})$

Deep network

$$\hat{\mathbf{y}} = \sigma\left(\mathbf{W}_3\ \sigma\left(\mathbf{W}_2\ \sigma\left(\mathbf{W}_1\mathbf{x}\right)\right)\right)$$

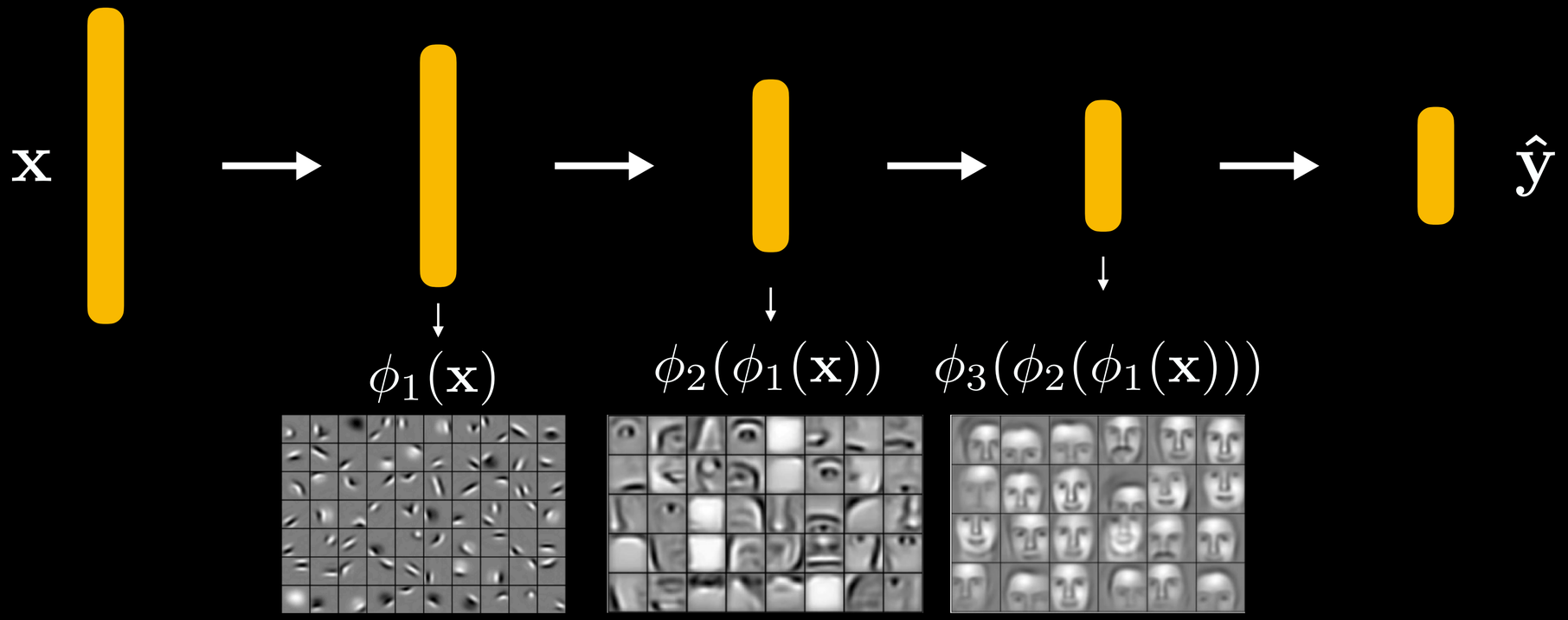


$$\mathbf{v}^{(1)} = \sigma\left(\mathbf{W}_1\mathbf{x}\right)$$

$$\mathbf{v}^{(2)} = \sigma\left(\mathbf{W}_2\mathbf{v}^{(1)}\right)$$

$$\hat{\mathbf{y}} = \sigma\left(\mathbf{W}_3\mathbf{v}^{(2)}\right)$$

Why *deep* learning?



Feature learning!

Lee, Honglak, et al. "Unsupervised learning of hierarchical representations with convolutional deep belief networks." Communications of the ACM 54.10 (2011)

Loss function

$$f_{\mathbf{W}_1 \mathbf{W}_2}(\mathbf{x}) = \sigma\left(\mathbf{W}_2\ \sigma\left(\mathbf{W}_1 \mathbf{x}\right)\right)$$

$$\mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{W}_1, \mathbf{W}_2) = \left\| f_{\mathbf{W}_1 \mathbf{W}_2}(\mathbf{x}) - \mathbf{y} \right\|^2$$

Stochastic gradient descent update

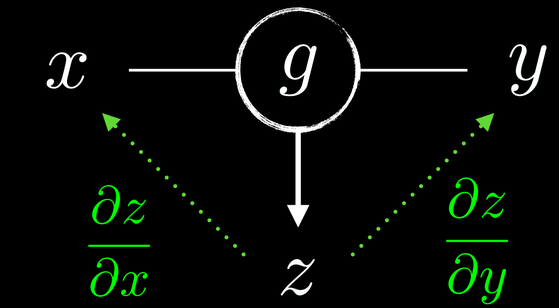
$$\mathbf{W}_1 \leftarrow \mathbf{W}_1 - \eta \nabla_{\mathbf{W}_1} \mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{W}_1, \mathbf{W}_2)$$

$$\mathbf{W}_2 \leftarrow \mathbf{W}_2 - \eta \nabla_{\mathbf{W}_2} \mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{W}_1, \mathbf{W}_2)$$

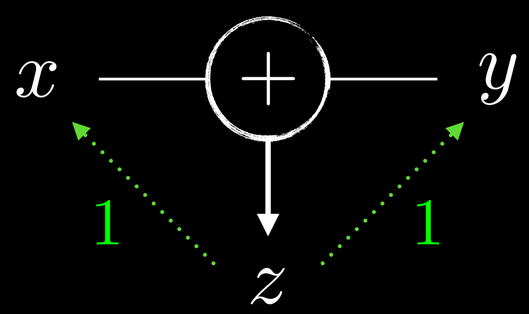
How do we calculate the gradients?

Computation graphs

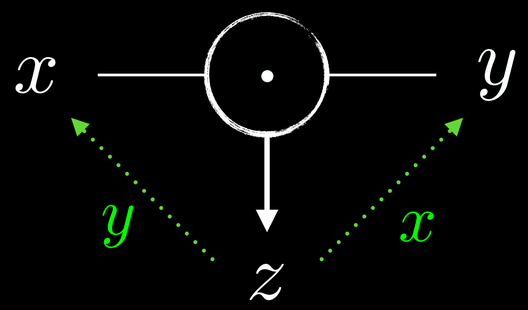
$$z = g(x, y)$$



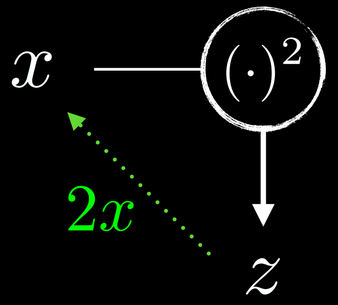
$$z = x + y$$



$$z = xy$$

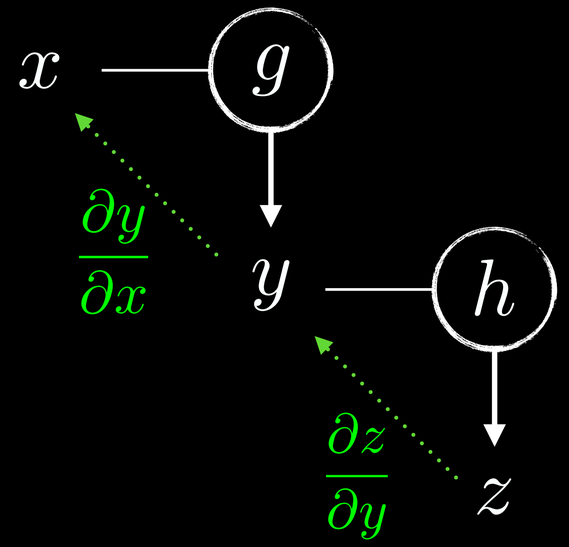


$$z = x^2$$



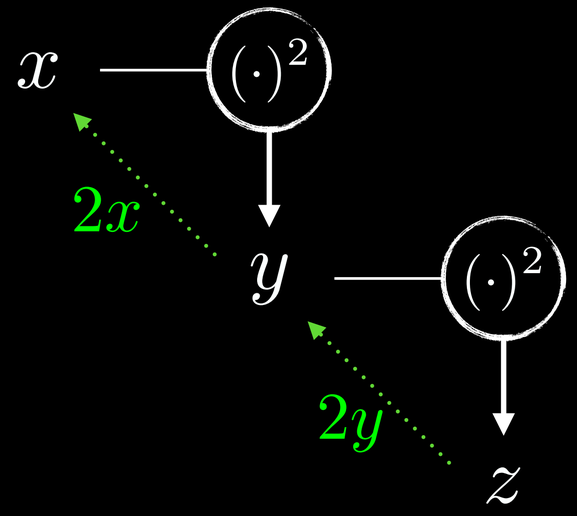
Chain rule

$$y = g(x)$$
$$z = h(y)$$



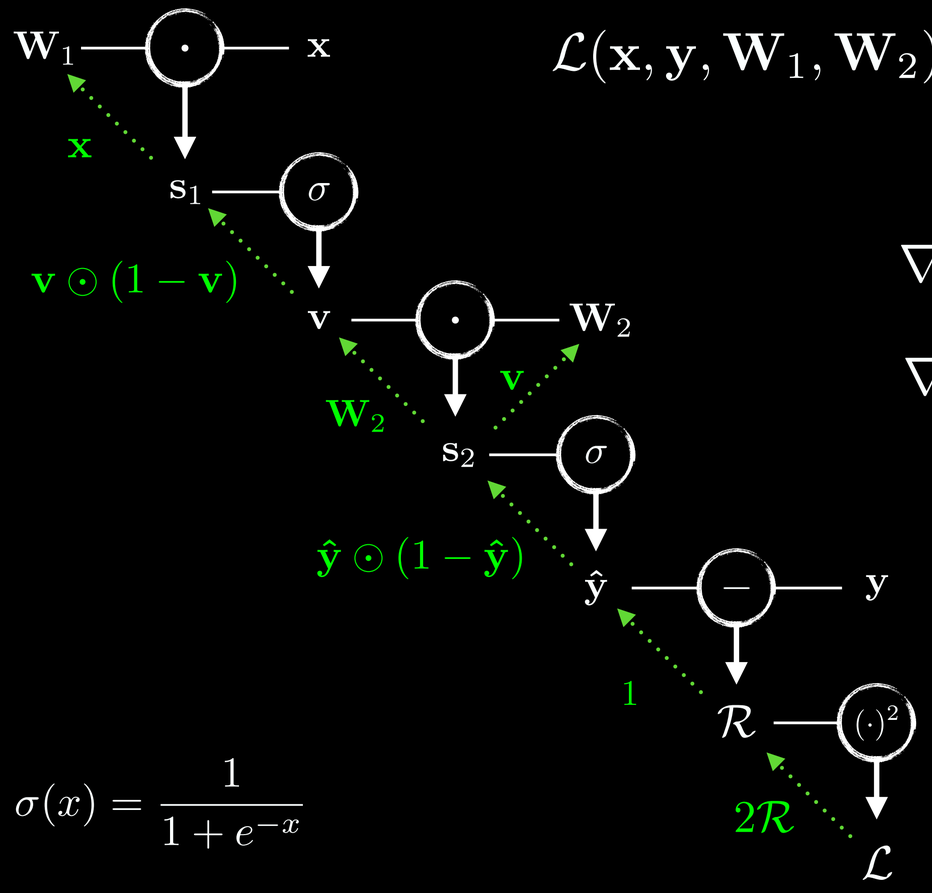
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$

$$y = x^2$$
$$z = y^2$$



$$\frac{\partial z}{\partial x} = 4xy = 4x^3$$

Backpropagation



$$\mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{W}_1, \mathbf{W}_2) = \|\sigma(\mathbf{W}_2\,\sigma(\mathbf{W}_1\mathbf{x})) - \mathbf{y}\|^2$$

$$\nabla_{\mathbf{W}_1}\mathcal{L} = 2\mathbf{W}_2^\top \mathcal{R} \odot \hat{\mathbf{y}} \odot (1 - \hat{\mathbf{y}}) \odot \mathbf{v} \odot (1 - \mathbf{v})\mathbf{x}^\top$$

$$\nabla_{\mathbf{W}_2}\mathcal{L} = 2\mathcal{R} \odot \hat{\mathbf{y}} \odot (1 - \hat{\mathbf{y}})\mathbf{v}^\top$$

assuming $\sigma(x) = \dfrac{1}{1 + e^{-x}}$

Optimization

Training loss

$$\mathcal{L}_{\text{train}}(\mathbf{W}_1, \mathbf{W}_2) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{D}_{\text{train}}} \mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{W}_1, \mathbf{W}_2)$$

Objective

$$\hat{\mathbf{W}}_1, \hat{\mathbf{W}}_2 = \underset{\mathbf{W}_1, \mathbf{W}_2}{\operatorname{argmin}} \mathcal{L}_{\text{train}}(\mathbf{W}_1, \mathbf{W}_2)$$

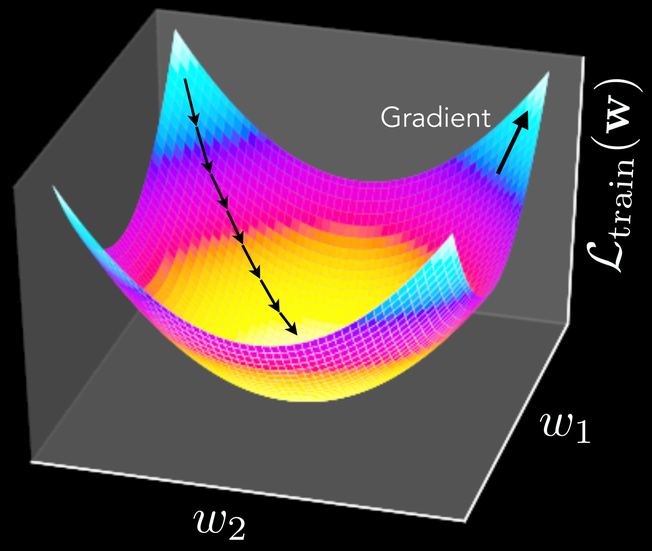
Optimal predictor

$$f_{\hat{\mathbf{W}}_1 \hat{\mathbf{W}}_2}(\mathbf{x}) = \sigma\left(\hat{\mathbf{W}}_2 \sigma\left(\hat{\mathbf{W}}_1 \mathbf{x}\right)\right)$$

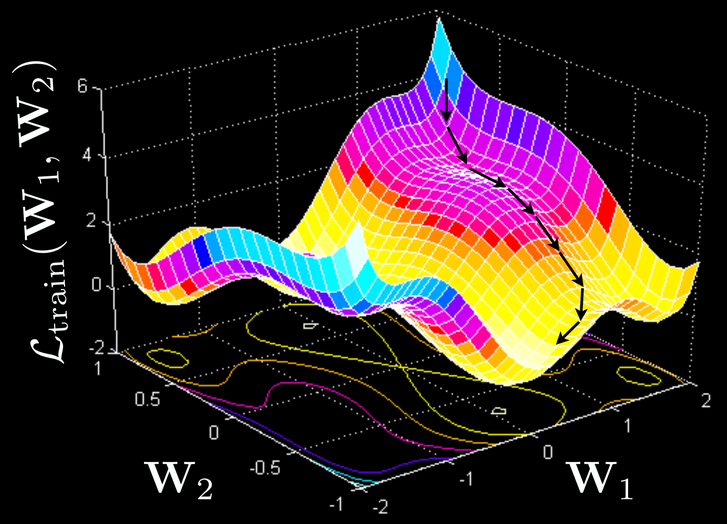
Non-convexity

$$\hat{\mathbf{W}}_1, \hat{\mathbf{W}}_2 = \underset{\mathbf{W}_1, \mathbf{W}_2}{\operatorname{argmin}} \mathcal{L}_{\text{train}}(\mathbf{W}_1, \mathbf{W}_2)$$

Linear predictor loss



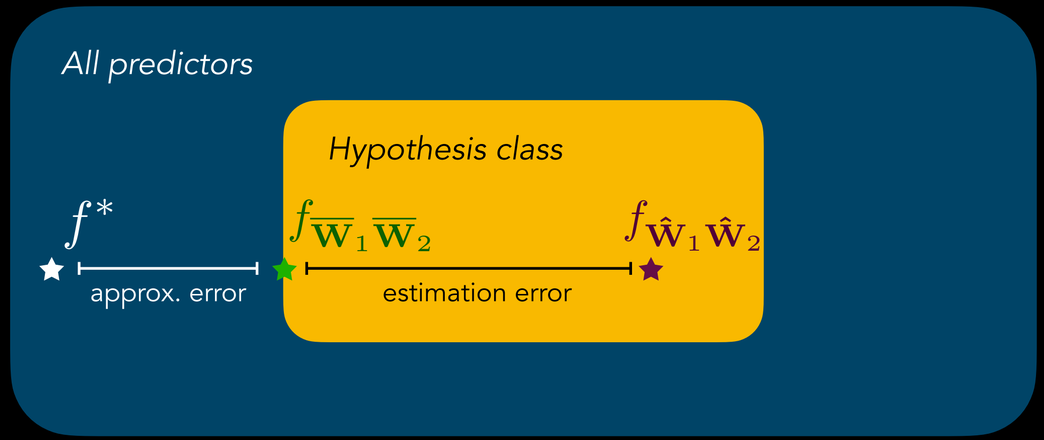
Neural network loss



Hypothesis Class

$$f_{\mathbf{W}_1 \mathbf{W}_2}(\mathbf{x}) = \sigma\left(\mathbf{W}_2\ \sigma\left(\mathbf{W}_1 \mathbf{x}\right)\right)$$

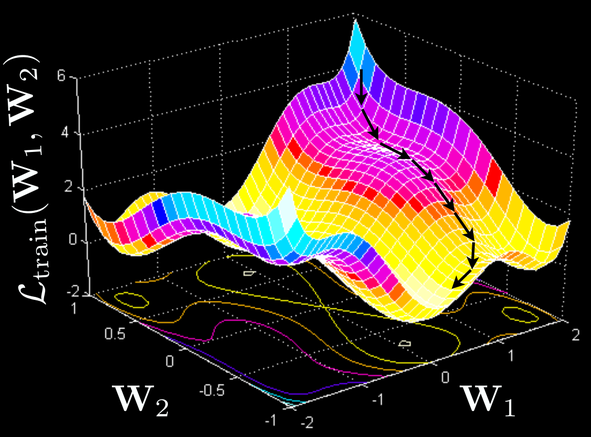
$$\mathcal{F} = \left\{ f_{\mathbf{W}_1 \mathbf{W}_2}(\mathbf{x}) \mid \mathbf{W}_1 \in \mathbb{R}^{k \times n}, \mathbf{W}_2 \in \mathbb{R}^{m \times k} \right\}$$



Hyperparameters

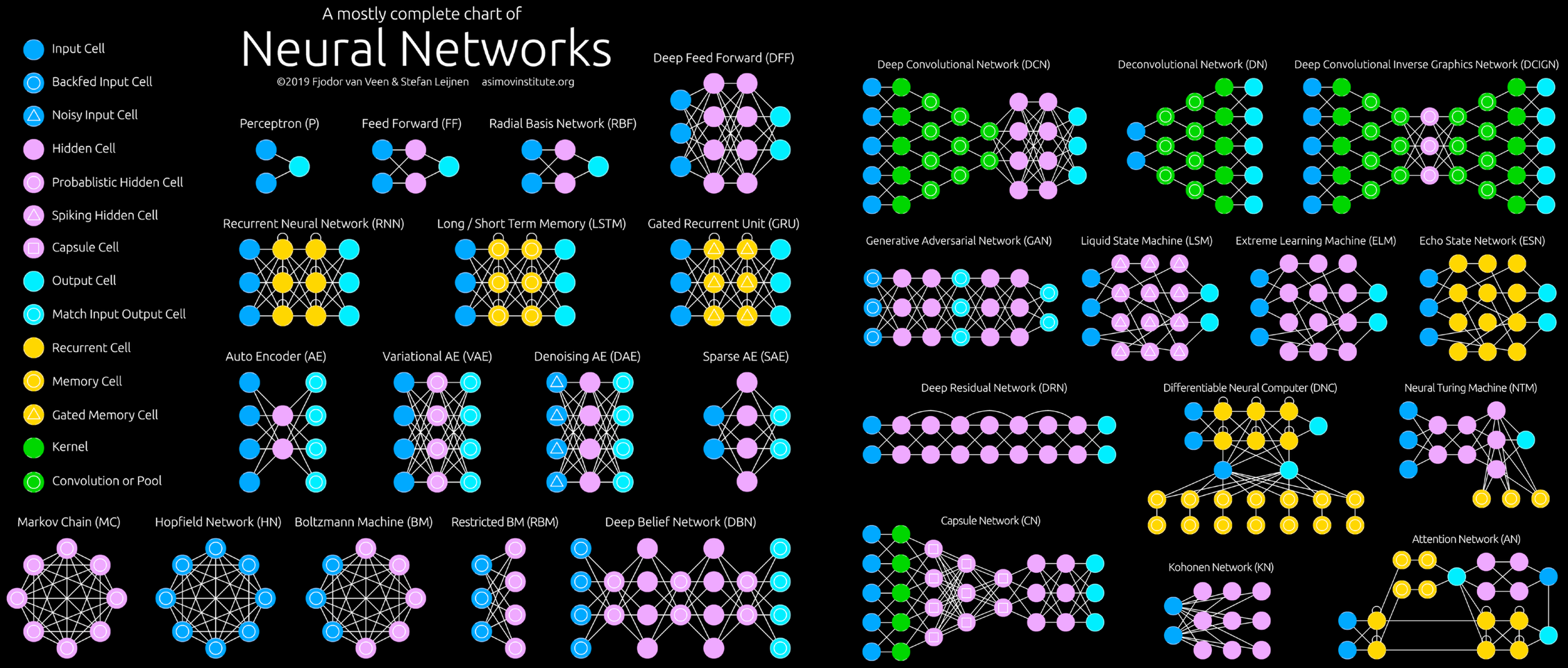
How do you train a deep network?

- Use many hidden layers for abstraction
- Use adaptive time steps
- Use hyper-parameter optimization



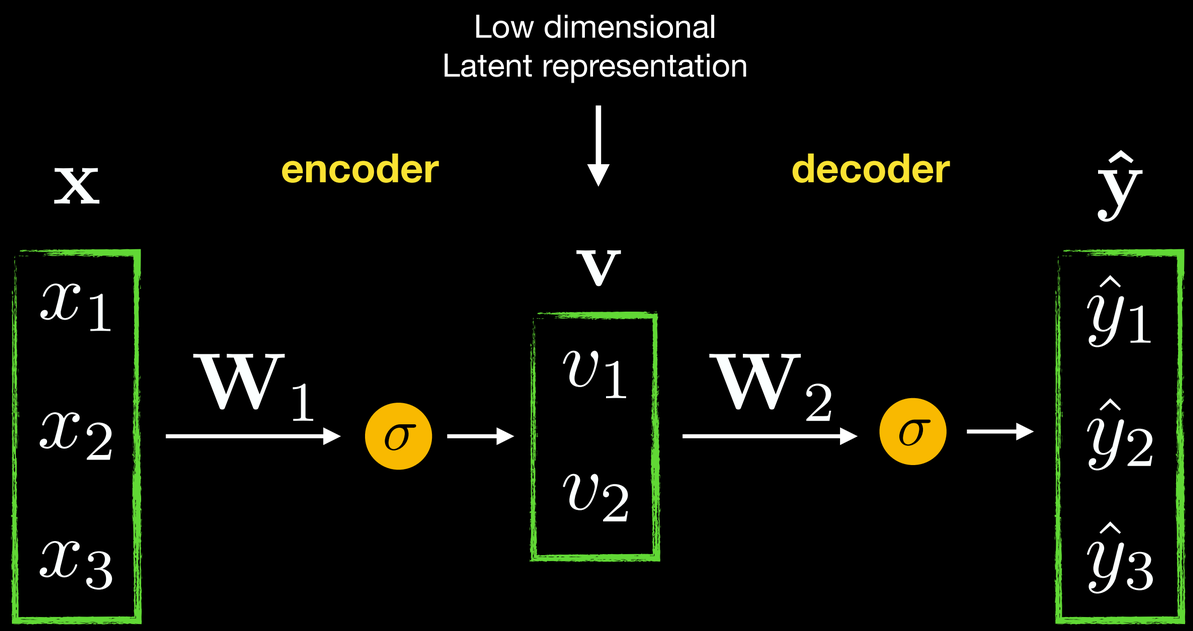
https://playground.tensorflow.org/

Neural network zoo



asimovinstitute.org

Auto-encoders

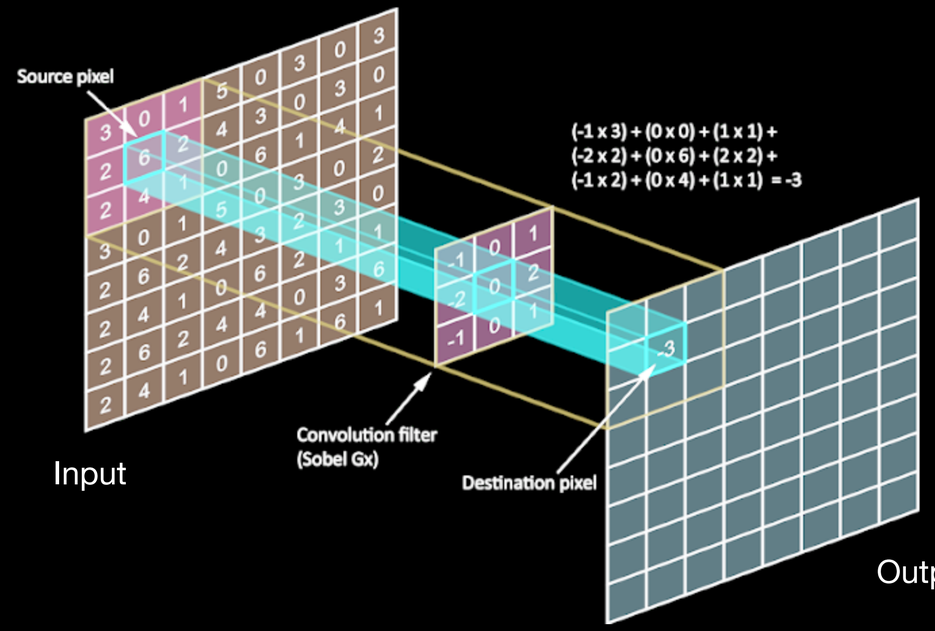


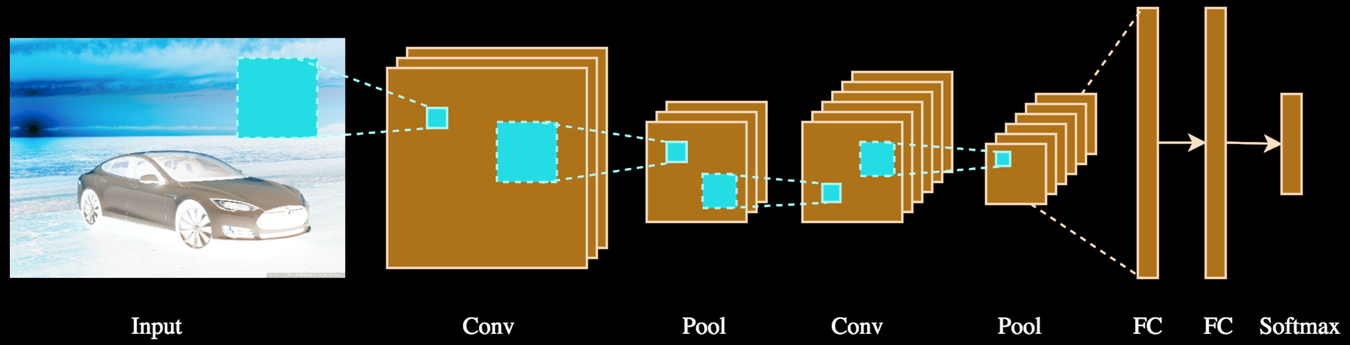
$$\mathcal{L} = \|f_{\mathbf{W}_1\mathbf{W}_2}(\mathbf{x}) - \mathbf{x}\|^2$$

if $\sigma = I$, network performs SVD decomposition

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$$

Convolutional neural networks



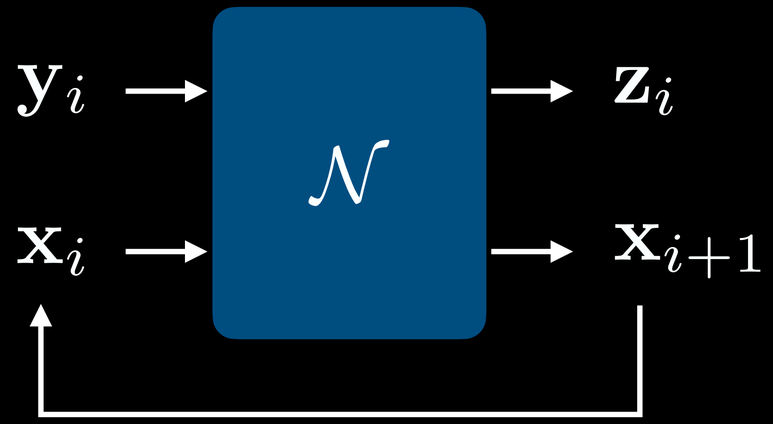


https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Recurrent neural networks

$$\boxed{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_i, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_n}$$

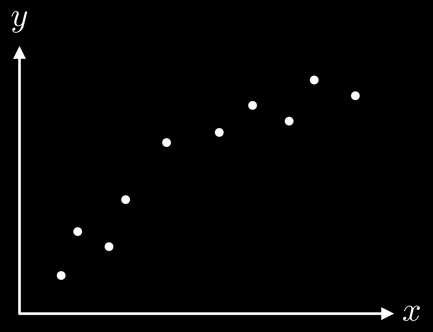
$$[\mathbf{x}_{i+1}, \mathbf{z}_i] = \mathcal{N}(\mathbf{x}_i, \mathbf{y}_i)$$



Feedback output to input

Numerical solvers are recurrence relations!

The ML workflow



- Data cleaning
- Dimensionality reduction
- Unsupervised metrics

$\phi(\mathbf{x})$

η

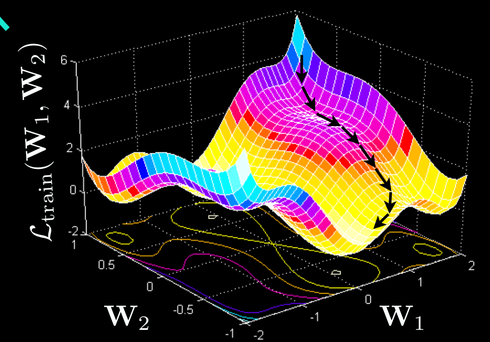
T λ

$\mathcal{L}_{\text{val}}$

not satisfied

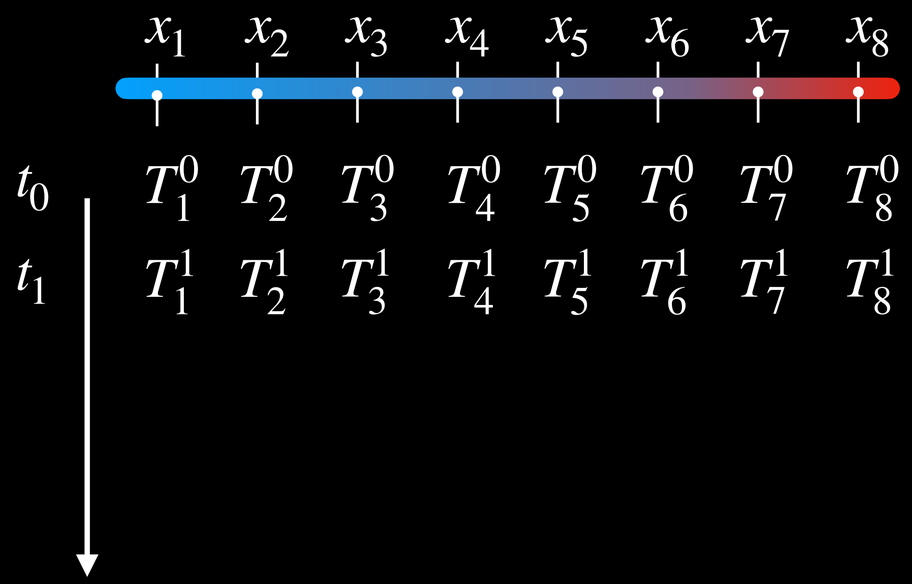
satisfied

$\mathcal{L}_{\text{test}}$



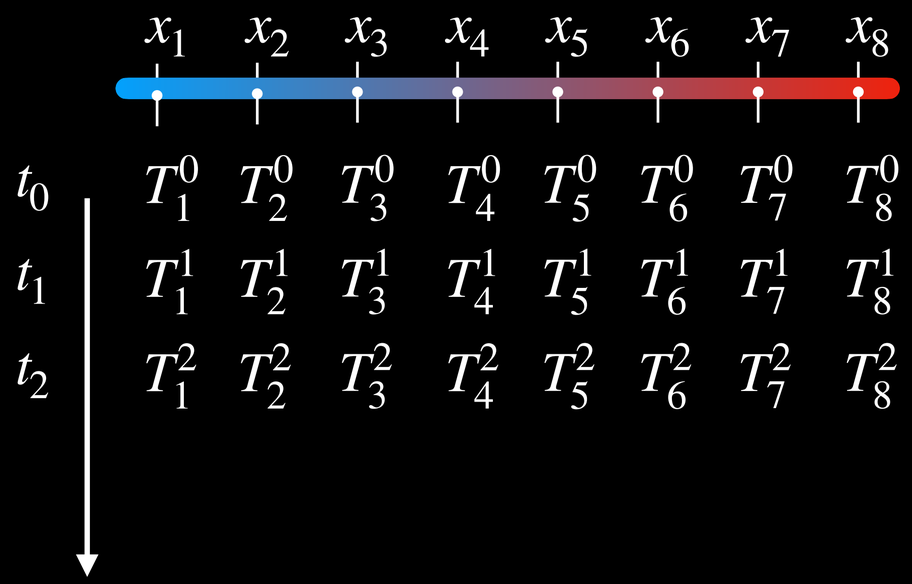
Derivative features

$$T_i^j = T(x_i, t_j)$$



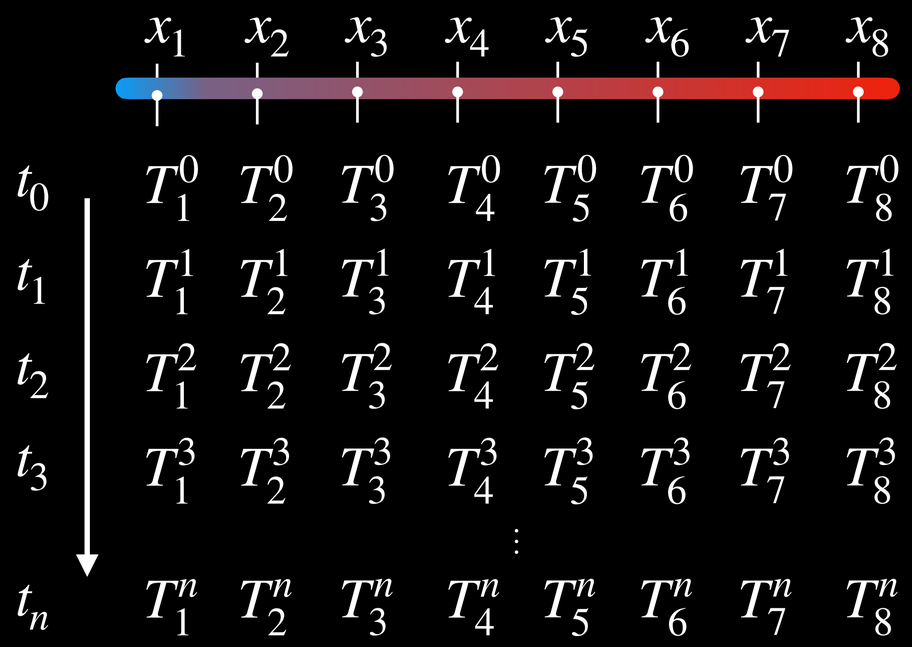
Derivative features

$$T_i^j = T(x_i, t_j)$$



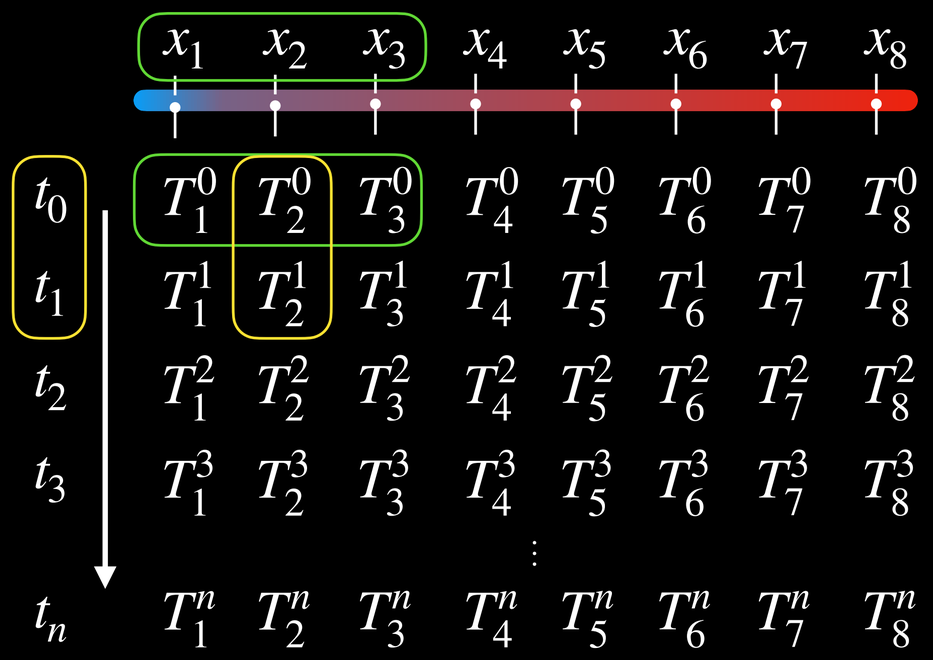
Derivative features

$$T_i^j = T(x_i, t_j)$$



Derivative features

$$T_i^j = T(x_i, t_j)$$



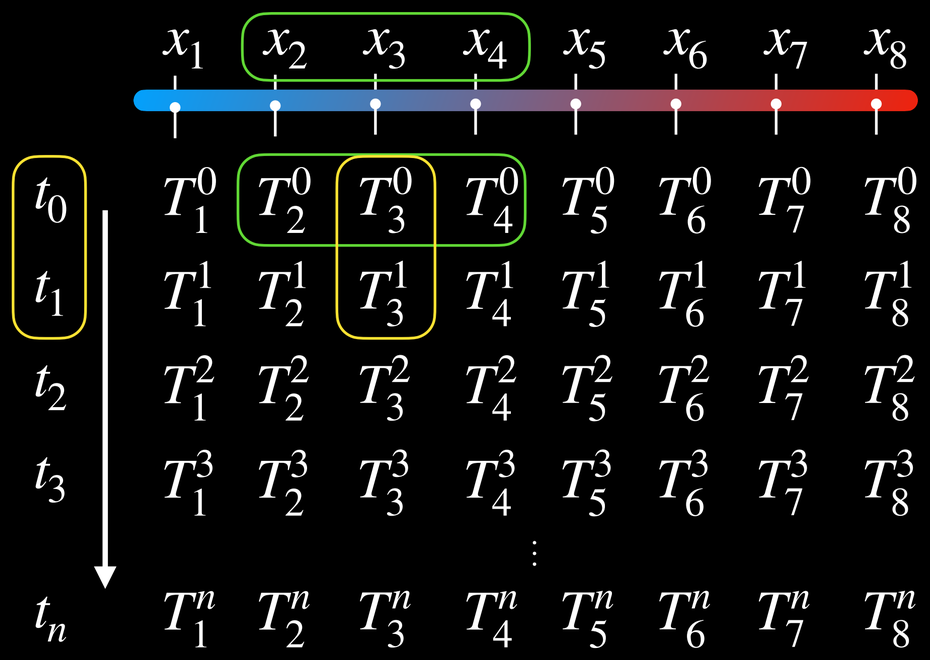
ϕ

$$\frac{T_i^{j+1} - T_i^j}{t_{j+1} - t_j} \qquad \frac{T_{i+1}^j - 2T_i^j + T_{i-1}^j}{x_{i+1} - 2x_i + x_{i-1}}$$

$$\left(\frac{\Delta T}{\Delta t}\right)_1 \qquad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_1$$

Derivative features

$$T_i^j = T(x_i, t_j)$$



ϕ

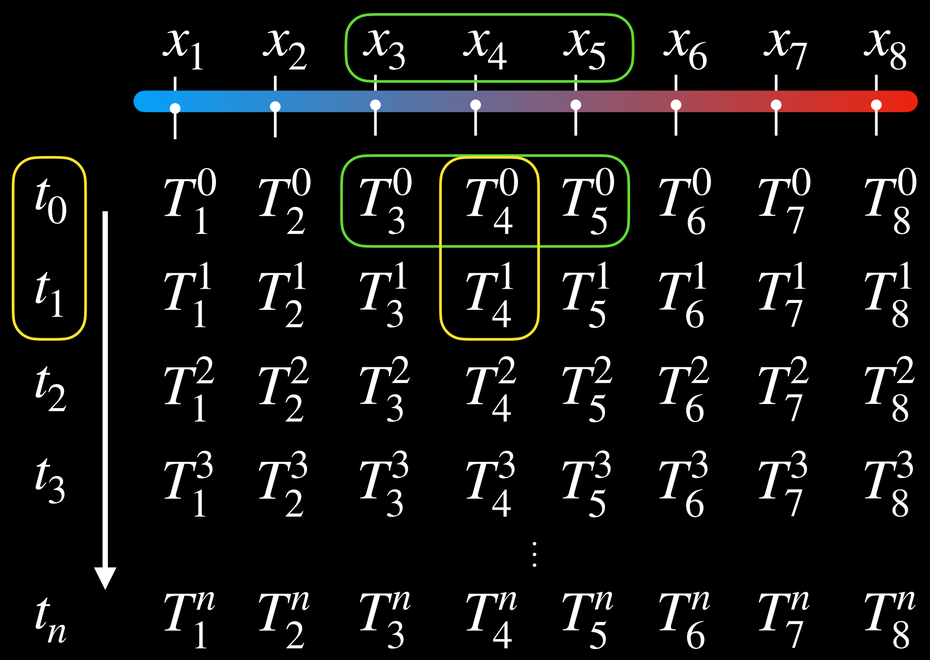
$$\frac{T_i^{j+1} - T_i^j}{t_{j+1} - t_j} \qquad \frac{T_{i+1}^j - 2T_i^j + T_{i-1}^j}{x_{i+1} - 2x_i + x_{i-1}}$$

$$\left(\frac{\Delta T}{\Delta t}\right)_1 \qquad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_1$$

$$\left(\frac{\Delta T}{\Delta t}\right)_2 \qquad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_2$$

Derivative features

$$T_i^j = T(x_i, t_j)$$



ϕ

$\frac{T_i^{j+1} - T_i^j}{t_{j+1} - t_j}$ $\quad$ $\frac{T_{i+1}^j - 2T_i^j + T_{i-1}^j}{x_{i+1} - 2x_i + x_{i-1}}$

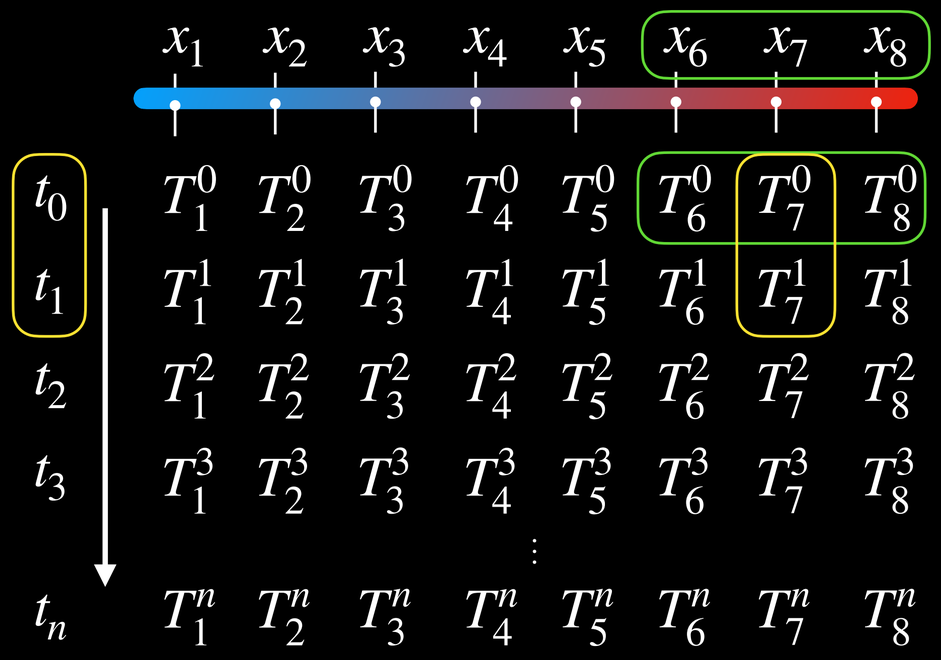
$\left(\frac{\Delta T}{\Delta t}\right)_1$ $\quad$ $\left(\frac{\Delta^2 T}{\Delta x^2}\right)_1$

$\left(\frac{\Delta T}{\Delta t}\right)_2$ $\quad$ $\left(\frac{\Delta^2 T}{\Delta x^2}\right)_2$

$\vdots$ $\quad$ $\vdots$

Derivative features

$$T_i^j = T(x_i, t_j)$$



ϕ

$$\frac{T_i^{j+1} - T_i^j}{t_{j+1} - t_j} \qquad \frac{T_{i+1}^j - 2T_i^j + T_{i-1}^j}{x_{i+1} - 2x_i + x_{i-1}}$$

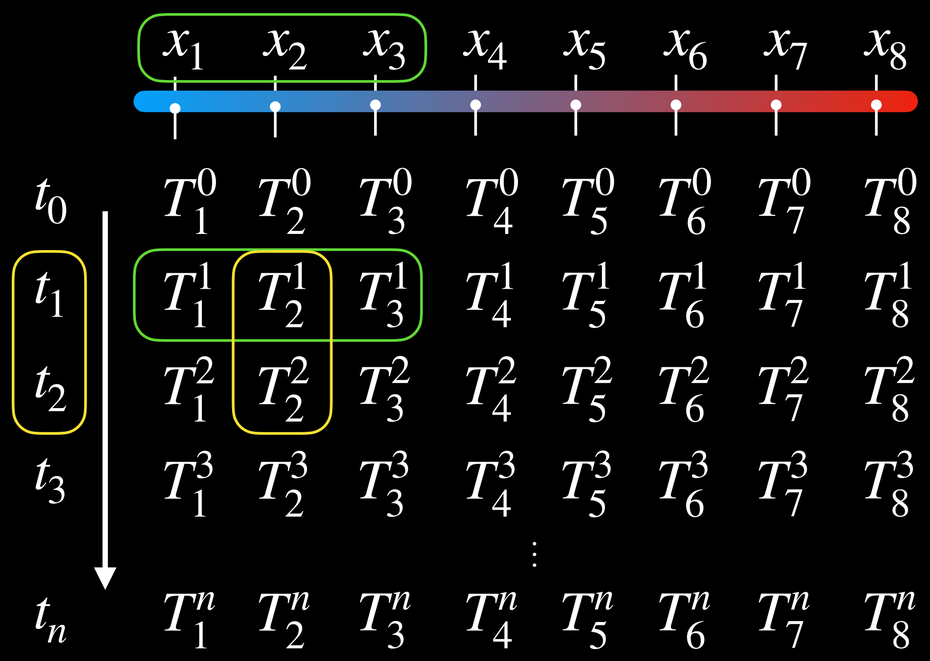
$$\left(\frac{\Delta T}{\Delta t}\right)_1 \qquad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_1$$

$$\left(\frac{\Delta T}{\Delta t}\right)_2 \qquad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_2$$

$$\vdots \qquad \vdots$$

Derivative features

$$T_i^j = T(x_i, t_j)$$



ϕ

$$\frac{T_i^{j+1} - T_i^j}{t_{j+1} - t_j} \qquad \frac{T_{i+1}^j - 2T_i^j + T_{i-1}^j}{x_{i+1} - 2x_i + x_{i-1}}$$

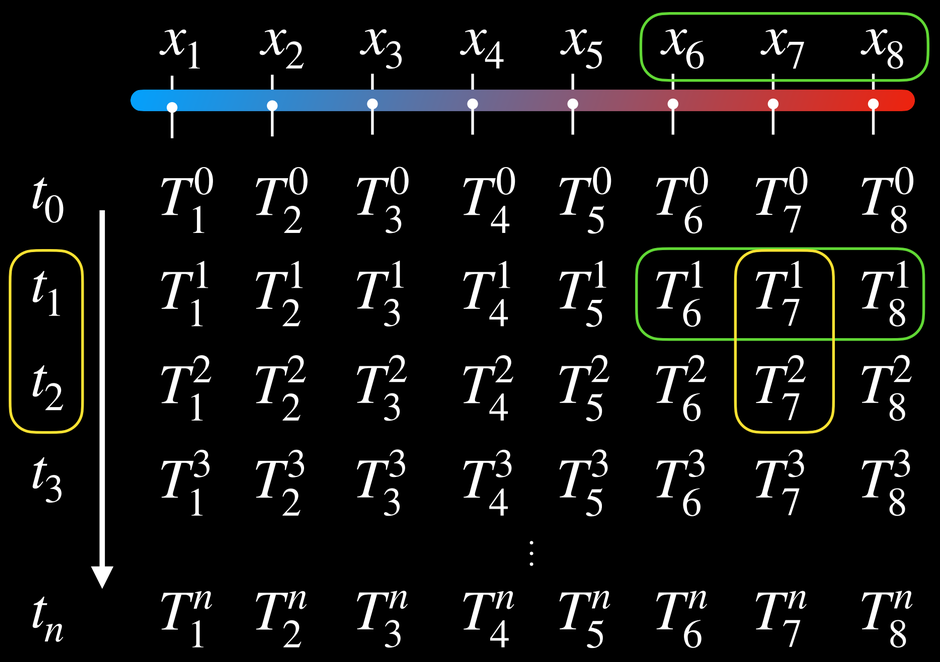
$$\left(\frac{\Delta T}{\Delta t}\right)_1 \qquad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_1$$

$$\left(\frac{\Delta T}{\Delta t}\right)_2 \qquad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_2$$

$\vdots$ $\vdots$

Derivative features

$$T_i^j = T(x_i, t_j)$$



ϕ

$$\frac{T_i^{j+1} - T_i^j}{t_{j+1} - t_j} \qquad \frac{T_{i+1}^j - 2T_i^j + T_{i-1}^j}{x_{i+1} - 2x_i + x_{i-1}}$$

$$\left(\frac{\Delta T}{\Delta t}\right)_1 \qquad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_1$$

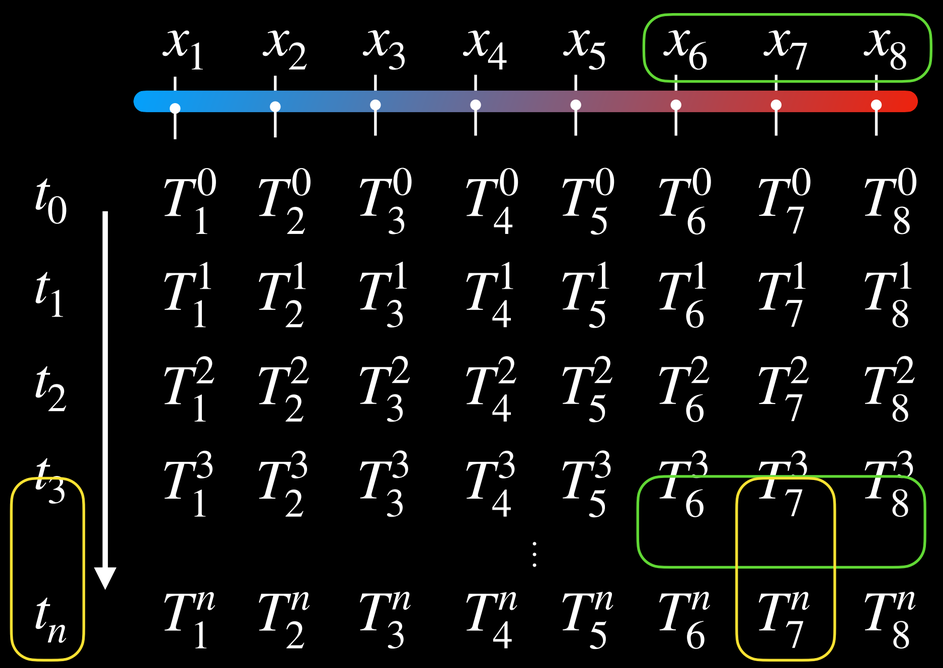
$$\left(\frac{\Delta T}{\Delta t}\right)_2 \qquad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_2$$

$$\vdots \qquad \vdots$$

$$\left(\frac{\Delta T}{\Delta t}\right)_n \qquad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_n$$

Derivative features

$$T_i^j = T(x_i, t_j)$$



ϕ

$$\frac{T_i^{j+1} - T_i^j}{t_{j+1} - t_j} \qquad \frac{T_{i+1}^j - 2T_i^j + T_{i-1}^j}{x_{i+1} - 2x_i + x_{i-1}}$$

$$\left(\frac{\Delta T}{\Delta t}\right)_1 \quad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_1$$

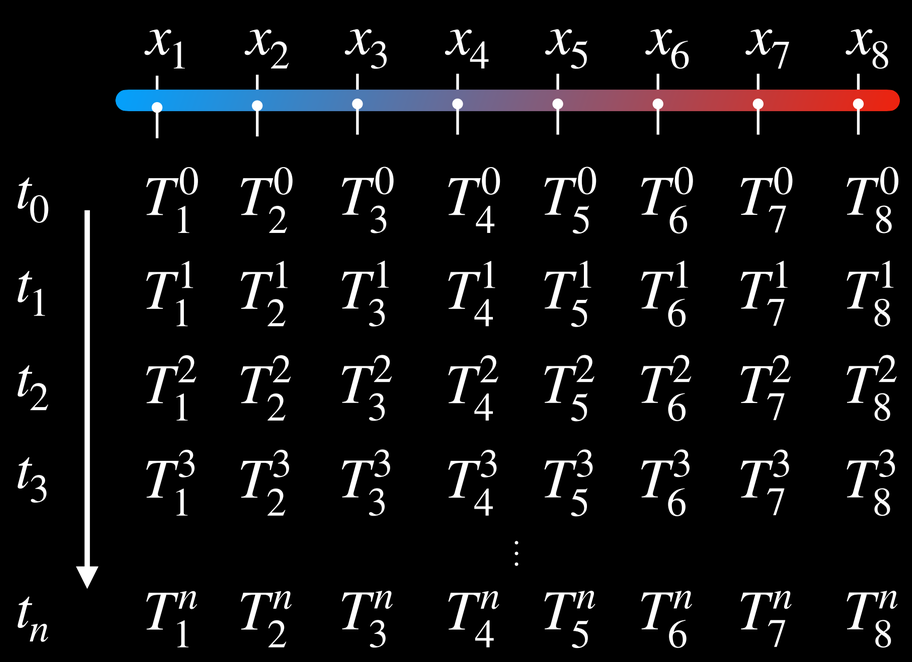
$$\left(\frac{\Delta T}{\Delta t}\right)_2 \quad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_2$$

$$\vdots \qquad \vdots$$

$$\left(\frac{\Delta T}{\Delta t}\right)_n \quad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_n$$

Derivative features

$$T_i^j = T(x_i, t_j)$$



ϕ

$$\frac{T_i^{j+1} - T_i^j}{t_{j+1} - t_j} \qquad \frac{T_{i+1}^j - 2T_i^j + T_{i-1}^j}{x_{i+1} - 2x_i + x_{i-1}}$$

$$\left(\frac{\Delta T}{\Delta t}\right)_1 \quad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_1$$

$$\left(\frac{\Delta T}{\Delta t}\right)_2 \quad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_2$$

$$\vdots \qquad \vdots$$

$$\left(\frac{\Delta T}{\Delta t}\right)_n \quad \left(\frac{\Delta^2 T}{\Delta x^2}\right)_n$$

Find w in: $\frac{\Delta T}{\Delta t} = w\frac{\Delta^2 T}{\Delta x^2}$