A jump-start into machine-learning

Dahvyd Wing

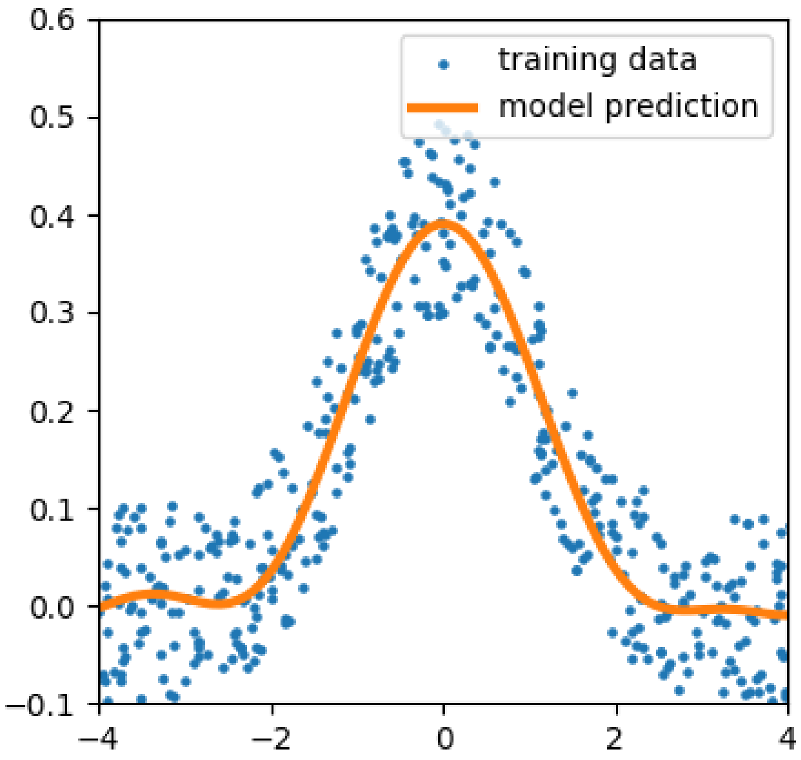
TCP

PHYMOL

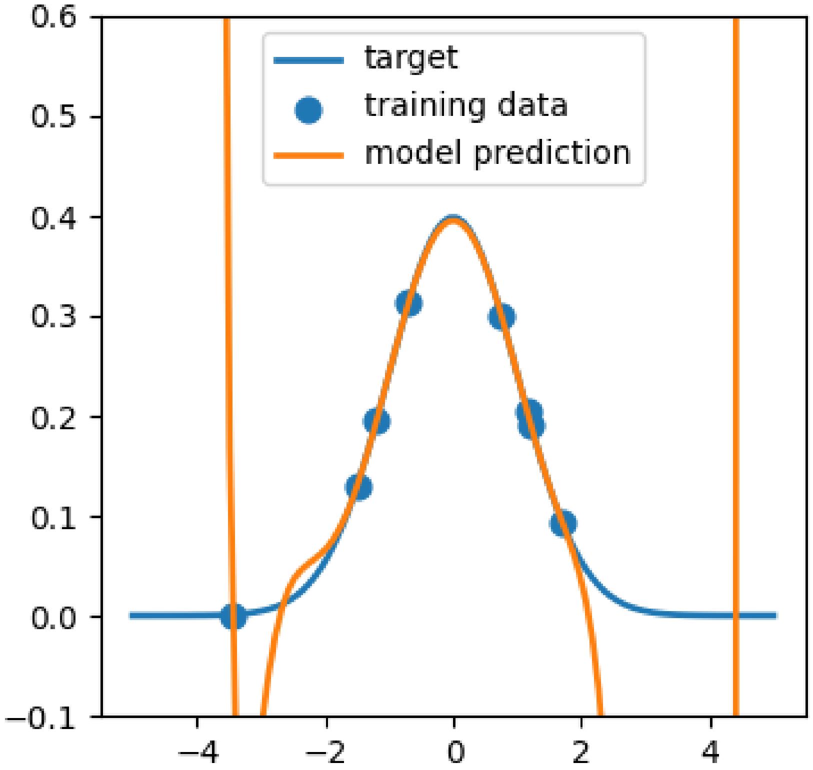
uni.lu
UNIVERSITÄT
LUXEMBURG

Regression for computational chemistry

Most applications: lots of noisy data



Our case: few data points with almost no noise



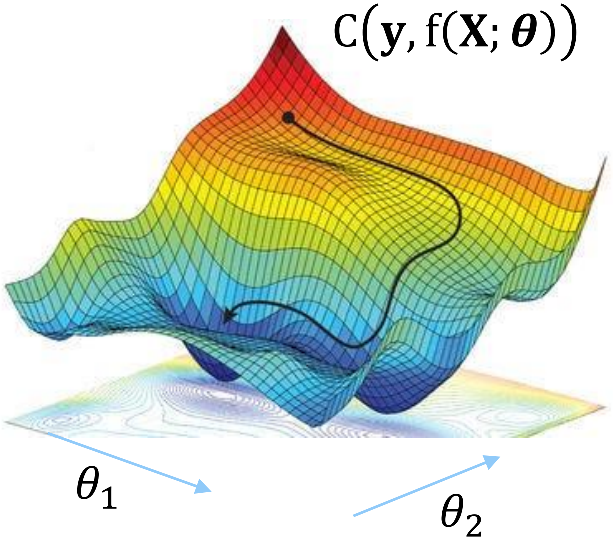
Anatomy of regression

1. Data: $(\boldsymbol{X}, \boldsymbol{y})$

2. Model: $f(\boldsymbol{x}; \boldsymbol{\theta})$
   $f: \boldsymbol{x} \rightarrow y$
   $\boldsymbol{\theta}$ are trainable parameters

3. Cost function: $\mathrm{C}\big(\mathbf{y}, \mathrm{f}(\mathbf{X}; \boldsymbol{\theta})\big)$
   mean squared error (MSE) $= \frac{1}{N}\sum_i\big(\boldsymbol{y}_i - f(\boldsymbol{x}_i, \boldsymbol{\theta})\big)^2$

4. Find $\min_{\boldsymbol{\theta}} \mathrm{C}\big(\mathbf{y}, \mathrm{f}(\mathbf{X}; \boldsymbol{\theta})\big)$ using gradient descent



Hutson, Science 2018

Pytorch example lj_1_overfit.py

1. Open anaconda prompt
2. conda activate ml_tutorial
3. Go to ml_tutorial folder
4. spyder &
5. In spyder open
   lj_1_overfit.py

Anatomy of regression

1. Data: $(\boldsymbol{X}, \boldsymbol{y})$
   - Instance/object of a customized dataset class
   - Implement 3 functions: __init__, __len__, and __get_item__
   - dataloader pulls random batches of data from the dataset

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x_1} \\ \boldsymbol{x_2} \\ \boldsymbol{x_3} \\ \vdots \end{pmatrix} \quad \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \end{pmatrix}$$

Anatomy of regression

1. Data: $(\boldsymbol{X}, \boldsymbol{y})$

2. Model: $f(\boldsymbol{x}; \boldsymbol{\theta})$
   - Instance/object of a customized nn.module class
   - Implement 2 functions: __init__ and __forward__

Anatomy of regression

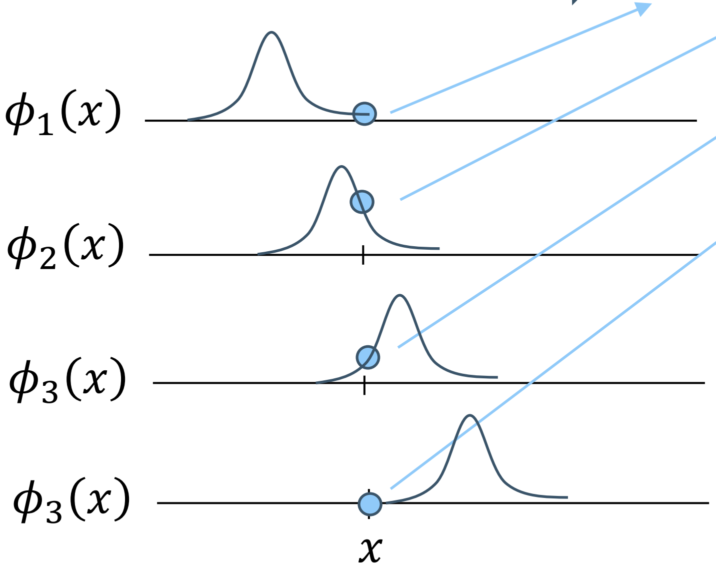
1. Data: $(\boldsymbol{X}, \boldsymbol{y})$

2. Model: $f(\boldsymbol{x}; \boldsymbol{\theta})$
   - Instance/object of a customized nn.module class
   - Implement 2 functions: __init__ and __forward__
   - Descriptor:

One hot encoding:

$$H = (1 \quad 0 \quad 0)$$
$$C = (0 \quad 1 \quad 0)$$
$$O = (0 \quad 0 \quad 1)$$

Continuous data: $x = 3.1$ ⇨ $(0.1 \quad 0.5 \quad 0.3 \quad 0.01 \quad 0 \quad 0)$



Anatomy of regression

1. Data: $(\boldsymbol{X}, \boldsymbol{y})$

2. Model: $f(\boldsymbol{x}; \boldsymbol{\theta})$
   - Instance/object of a customized nn.module class
   - Implement 2 functions: __init__ and __forward__
   - Descriptor
   - Neural network:

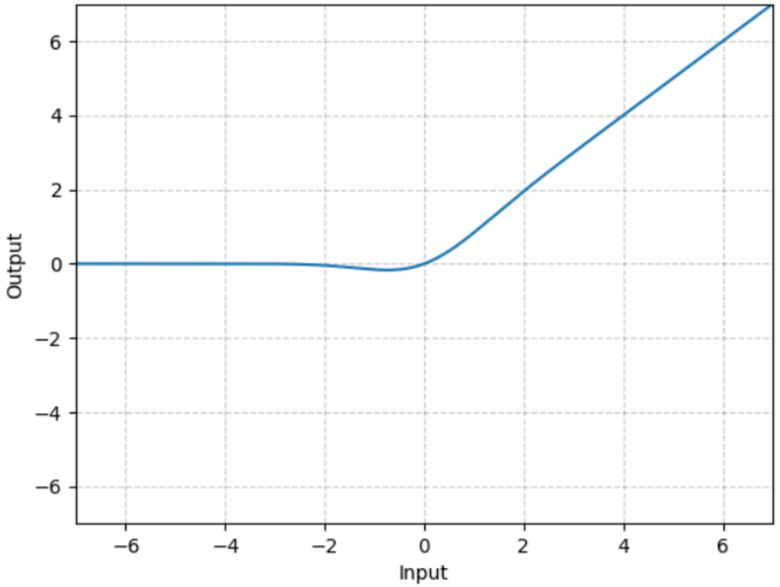
$$\boldsymbol{y_1} = \sigma(\boldsymbol{W_0 x} + \boldsymbol{b_0})$$

$$\boldsymbol{y_2} = \sigma(\boldsymbol{W_1 y_1} + \boldsymbol{b_1})$$

$$\boldsymbol{y_3} = \sigma(\boldsymbol{W_2 y_2} + \boldsymbol{b_2})$$

$$\mathrm{y}_{\mathrm{pred}} = \boldsymbol{w_3} \cdot \boldsymbol{y_3} + b_3$$

$\sigma(x)$ is the nonlinear activation function: GELU



Use a continuously differentiable activation function

Anatomy of regression

1. Data: $(\boldsymbol{X}, \boldsymbol{y})$

2. Model: $f(\boldsymbol{x}; \boldsymbol{\theta})$

3. Cost function: $\mathrm{C}\left(\mathbf{y}, \mathrm{f}(\mathbf{X}; \boldsymbol{\theta})\right)$
   - Mean squared error

Anatomy of regression

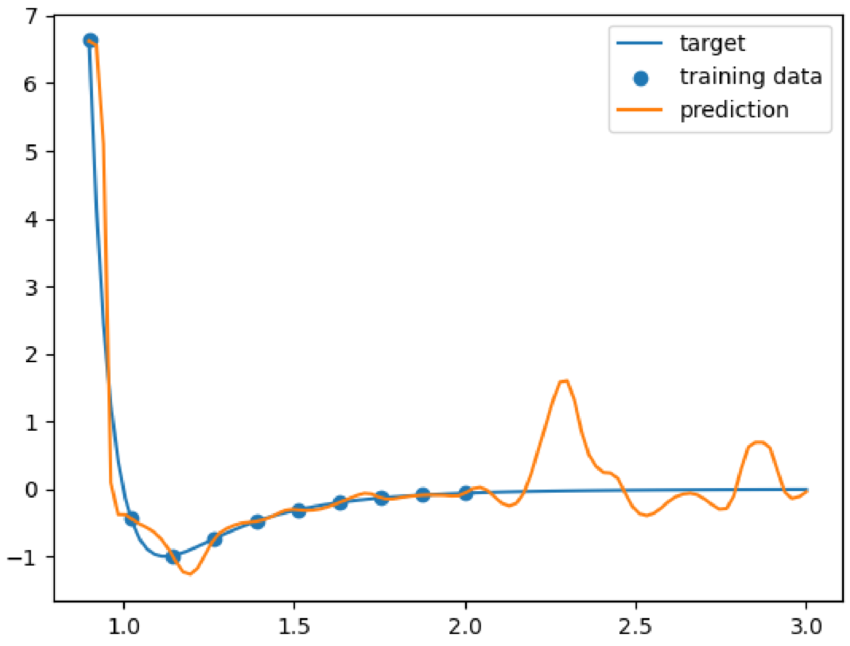
1. Data: $(\boldsymbol{X}, \boldsymbol{y})$
2. Model: $f(\boldsymbol{x}; \boldsymbol{\theta})$
3. Cost function: $\mathrm{C}\big(\mathbf{y}, \mathrm{f}(\mathbf{X}; \boldsymbol{\theta})\big)$
4. Find $\min_{\boldsymbol{\theta}} \mathrm{C}\big(\mathbf{y}, \mathrm{f}(\mathbf{X}; \boldsymbol{\theta})\big)$

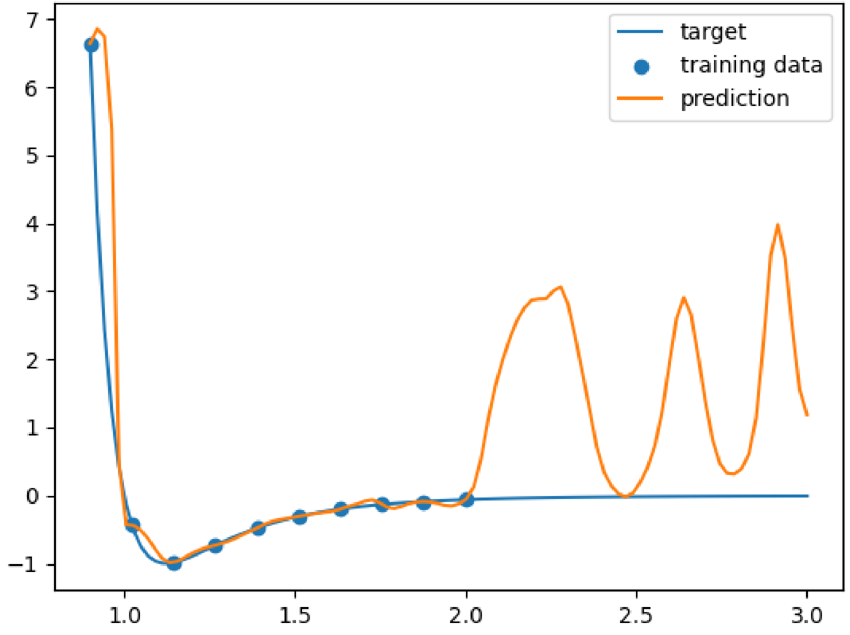
Batch 1	Batch 2	Batch 3	
(x_1, y_1), (x_3, y_3), (x_8, y_8)	(x_2, y_2), (x_5, y_5), (x_6, y_6)	(x_4, y_4), (x_7, y_7), (x_9, y_9)	$\cdots$
$\boldsymbol{\theta}' = \boldsymbol{\theta} - \nabla_{\boldsymbol{\theta}} \mathrm{C}\big(\mathbf{y}, \mathrm{f}(\mathbf{X}; \boldsymbol{\theta})\big)$	$\boldsymbol{\theta}' = \boldsymbol{\theta} - \nabla_{\boldsymbol{\theta}} \mathrm{C}\big(\mathbf{y}, \mathrm{f}(\mathbf{X}; \boldsymbol{\theta})\big)$	$\boldsymbol{\theta}' = \boldsymbol{\theta} - \nabla_{\boldsymbol{\theta}} \mathrm{C}\big(\mathbf{y}, \mathrm{f}(\mathbf{X}; \boldsymbol{\theta})\big)$	

1 Epoch

Overfitting

- NNs often have many more parameters than samples in the training data
- Run lj_1_overfit.py several times
  - 311 trainable parameters, 10 data points



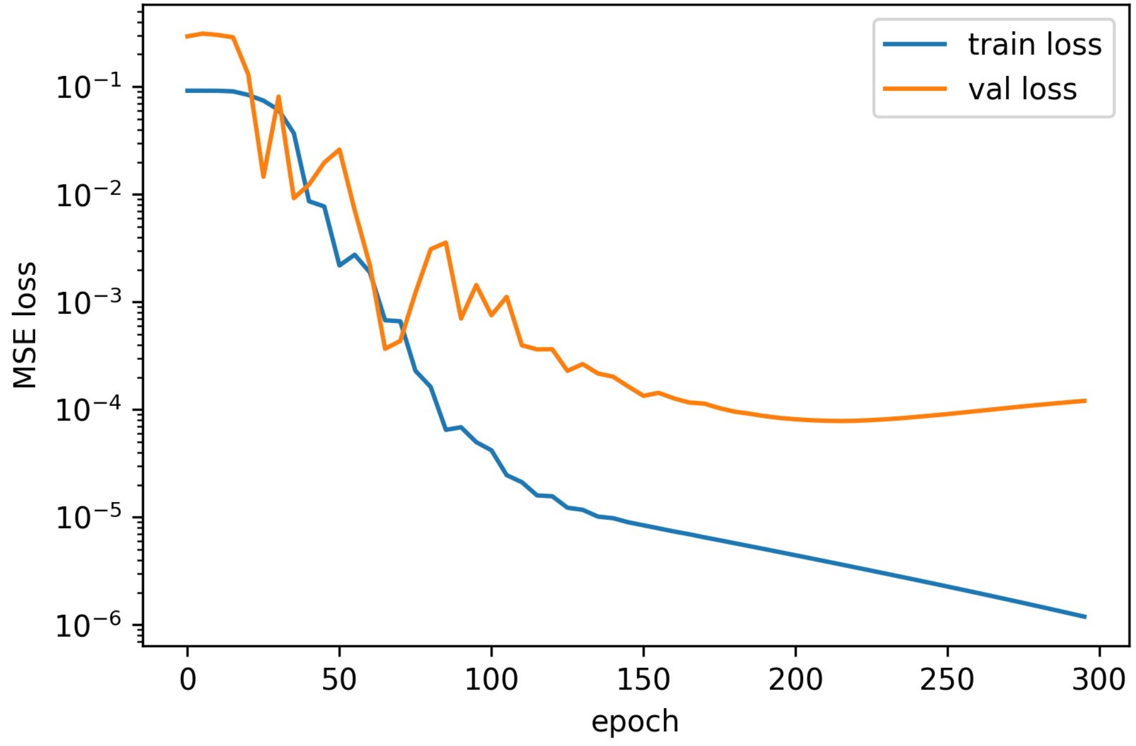


- Each model perfectly fits the training points, but doesn't do a great job in the interpolation region
- **You measure a model by testing on data it has never seen**
- The models do terrible in the extrapolation regime

Validation and test sets

- Separate your data into a training set, a validation set, and a test set
- Validation set used to measure overfitting and tune hyperparameters
- Test set is only used for the final model to get a final estimate of how accurate the model really is

- Run lj_2_overfit_with_validation.py
  - The main change in the code is lines 56-58

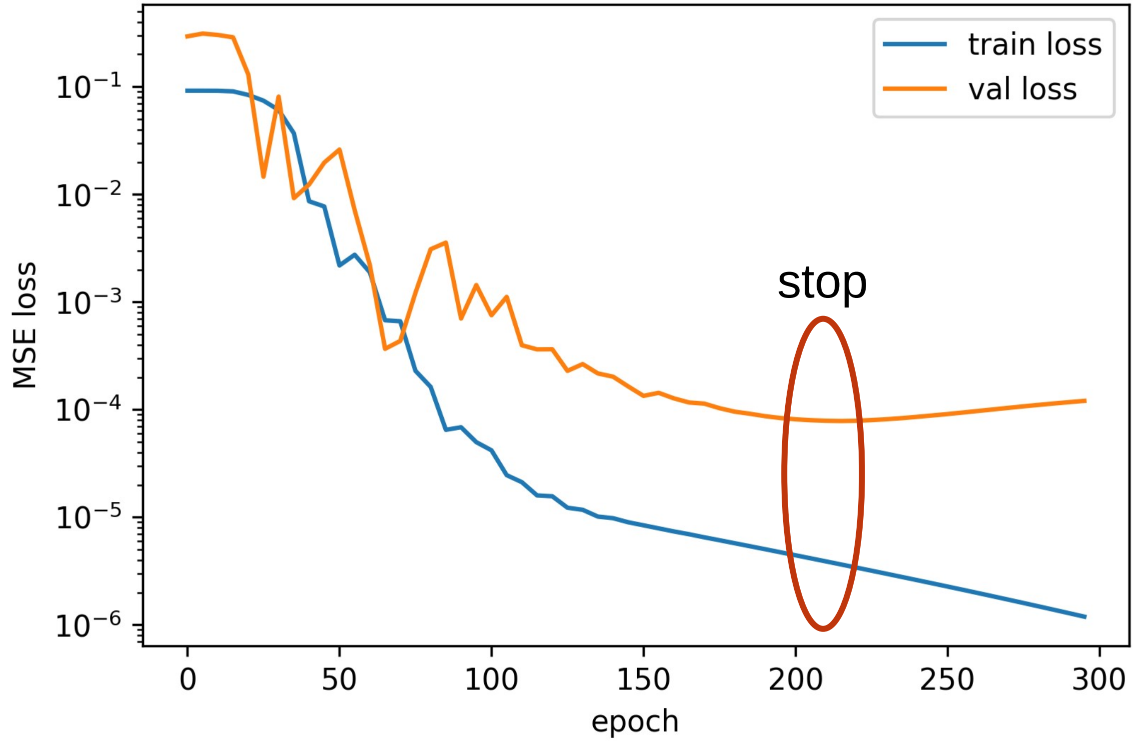


Validation and test sets

- Separate your data into a training set, a validation set, and a test set
- Validation set used to measure overfitting and tune hyperparameters
- Test set is only used for the final model to get a final estimate of how accurate the model really is

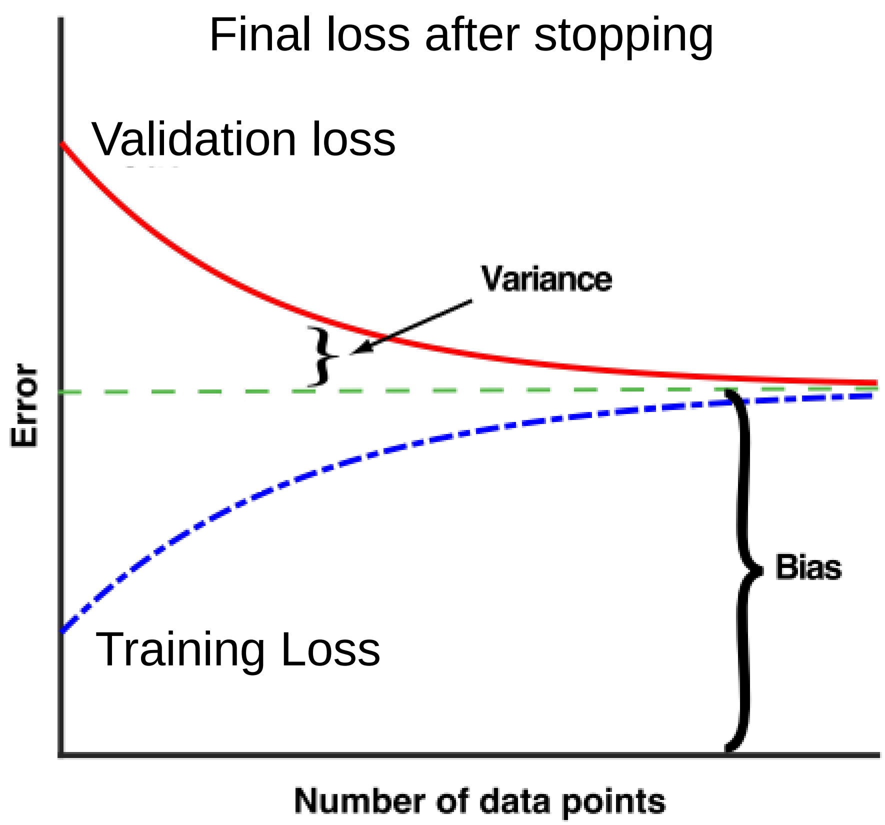
- Run lj_2_overfit_with_validation.py
  - The main change in the code is lines 56-58

- To get the best performance model stop when there is a steady increase in validation loss and decrease in training loss (early stopping)



Learning curve

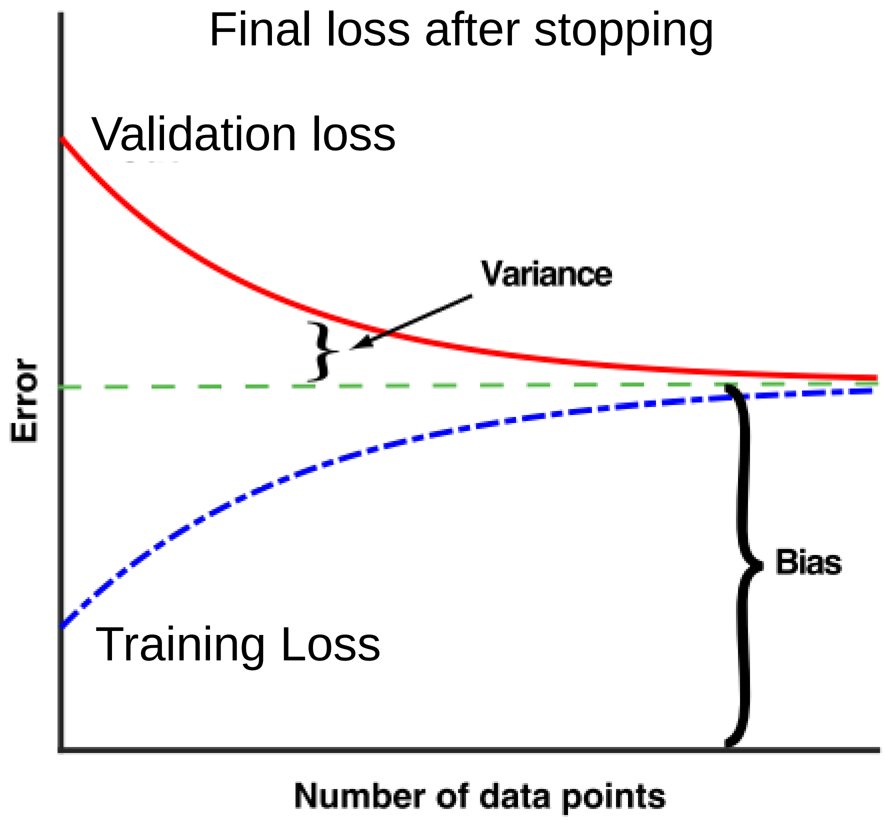
- With enough data the validation loss and training loss should converge

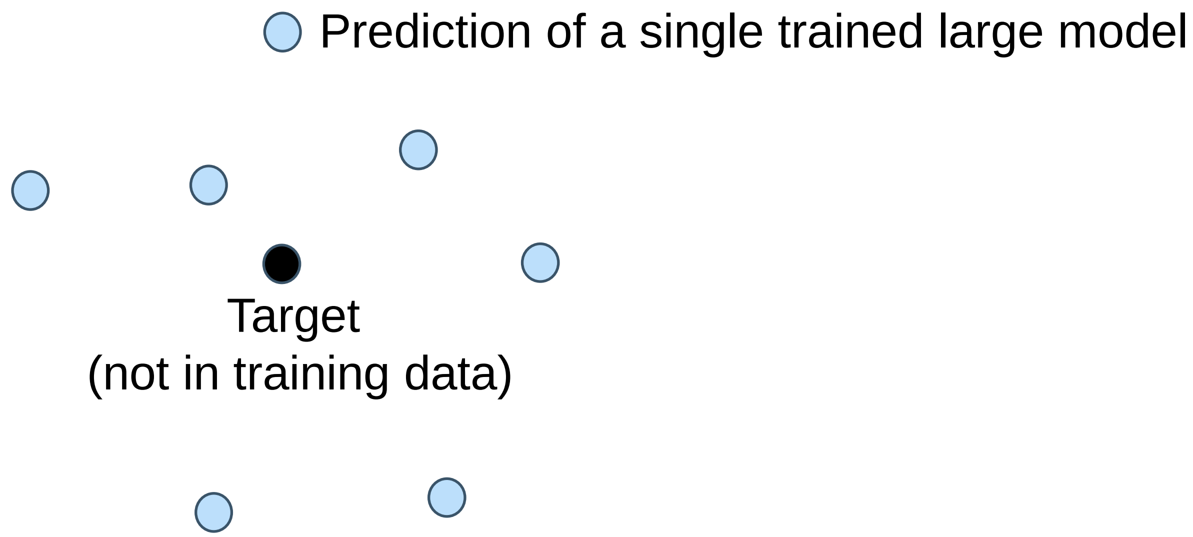


Mehta et al. Phys. Rep. **810** (2019)

Learning curve

- With enough data the validation loss and training loss should converge
- Variance: a models trained with different, but equal number of points yield different results
  - The more parameters, the more variance in the model

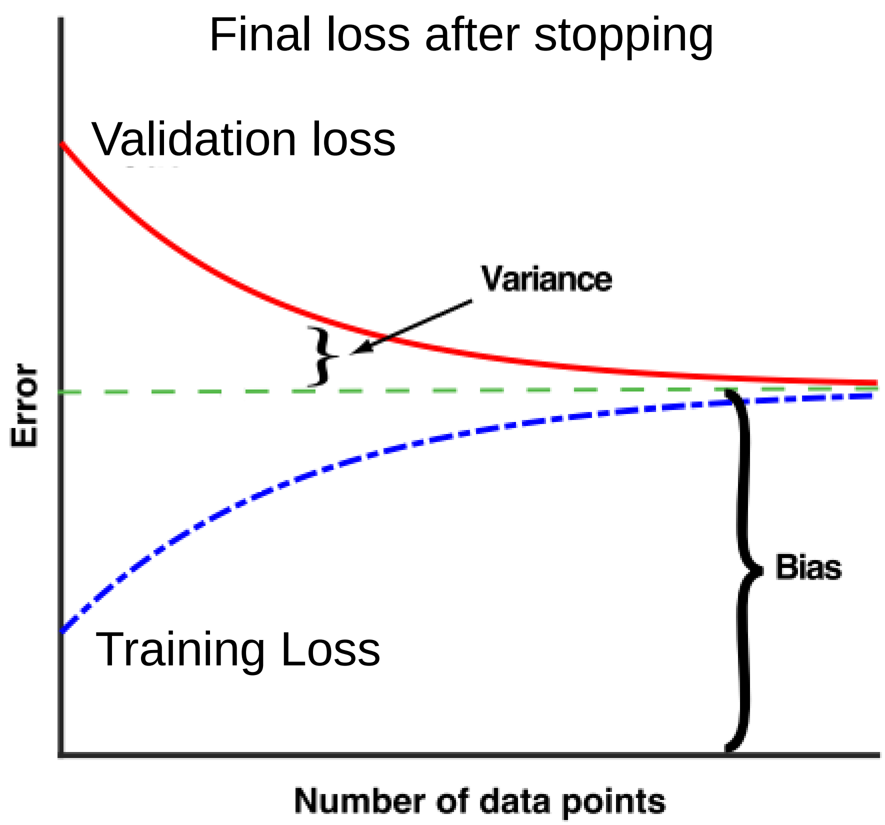


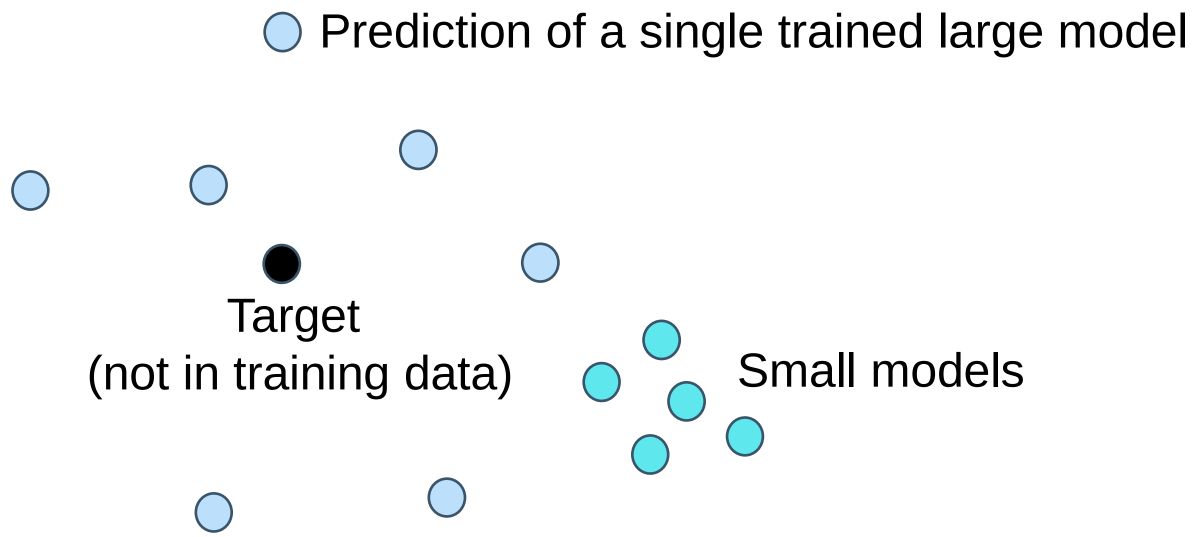


Mehta et al. Phys. Rep. **810** (2019)

Learning curve

- With enough data the validation loss and training loss should converge
- Variance: a models trained with different, but equal number of points yield different results
  - The more parameters, the more variance in the model

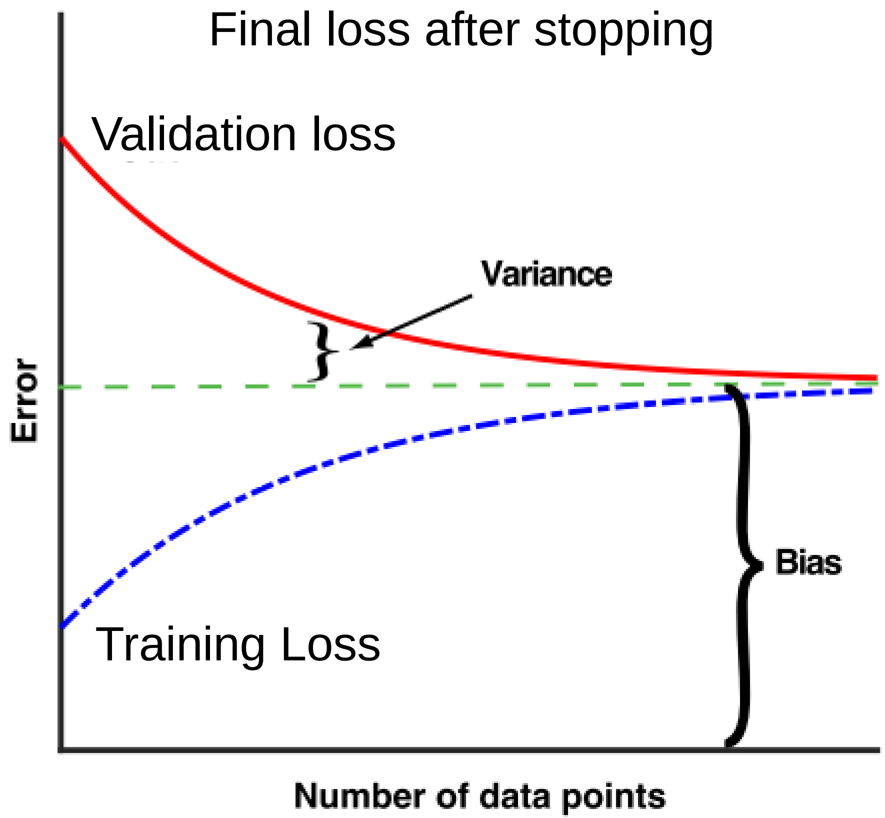


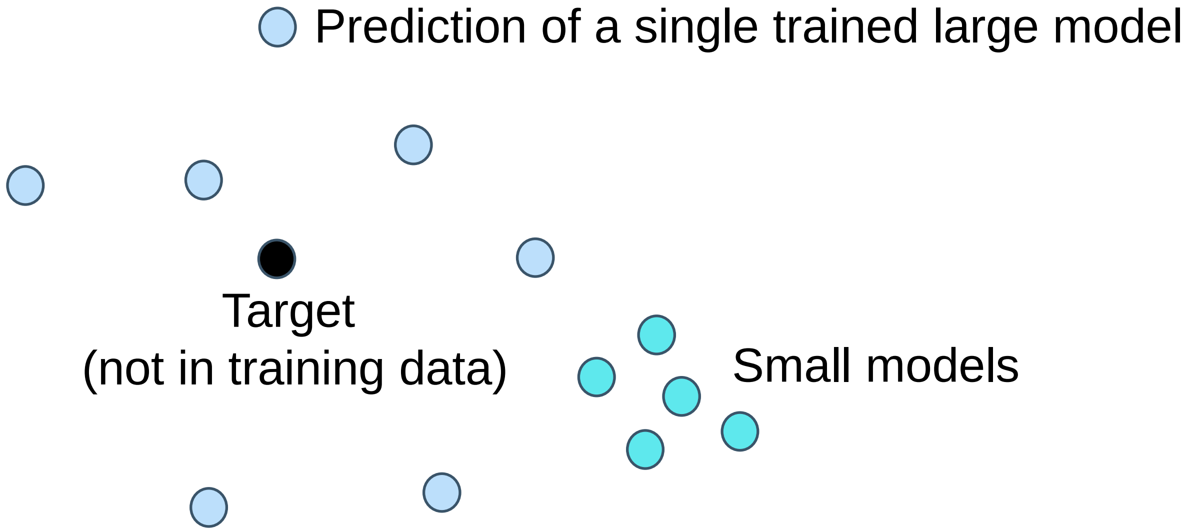


Mehta et al. Phys. Rep. **810** (2019)

Learning curve

- With enough data the validation loss and training loss should converge
- Variance: a models trained with different, but equal number of points yield different results
  - The more parameters, the more variance in the model
- Large high variance models overfit

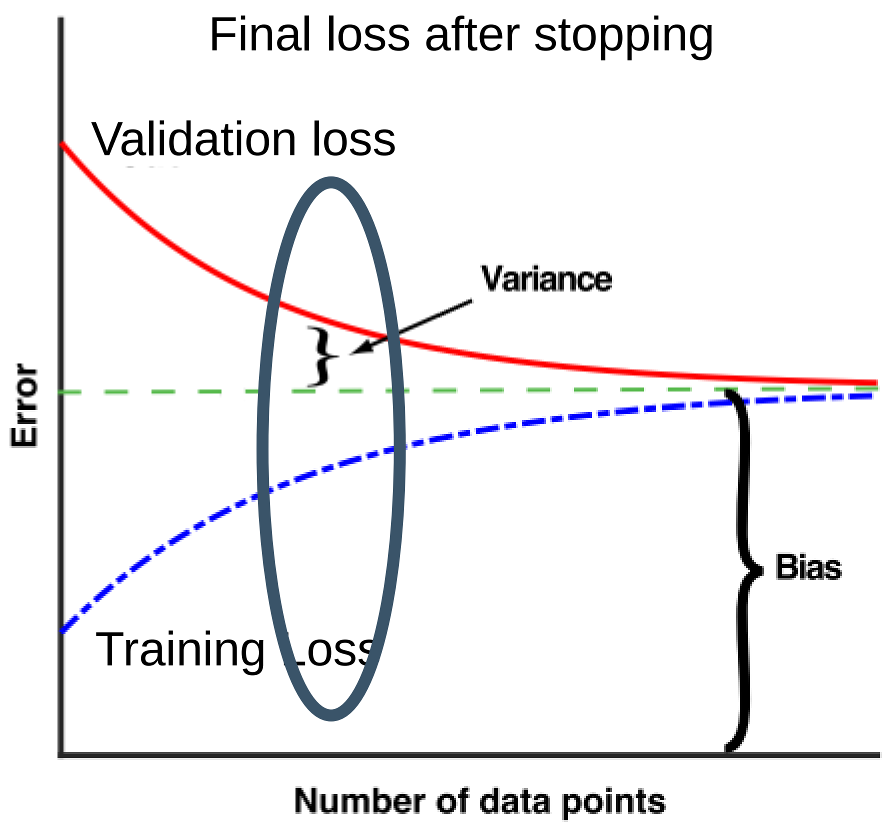




Mehta et al. Phys. Rep. **810** (2019)

Learning curve

- With enough data the validation loss and training loss should converge
- Variance: a models trained with different, but equal number of points yield different results
  - The more parameters, the more variance in the model
- Large high variance models overfit

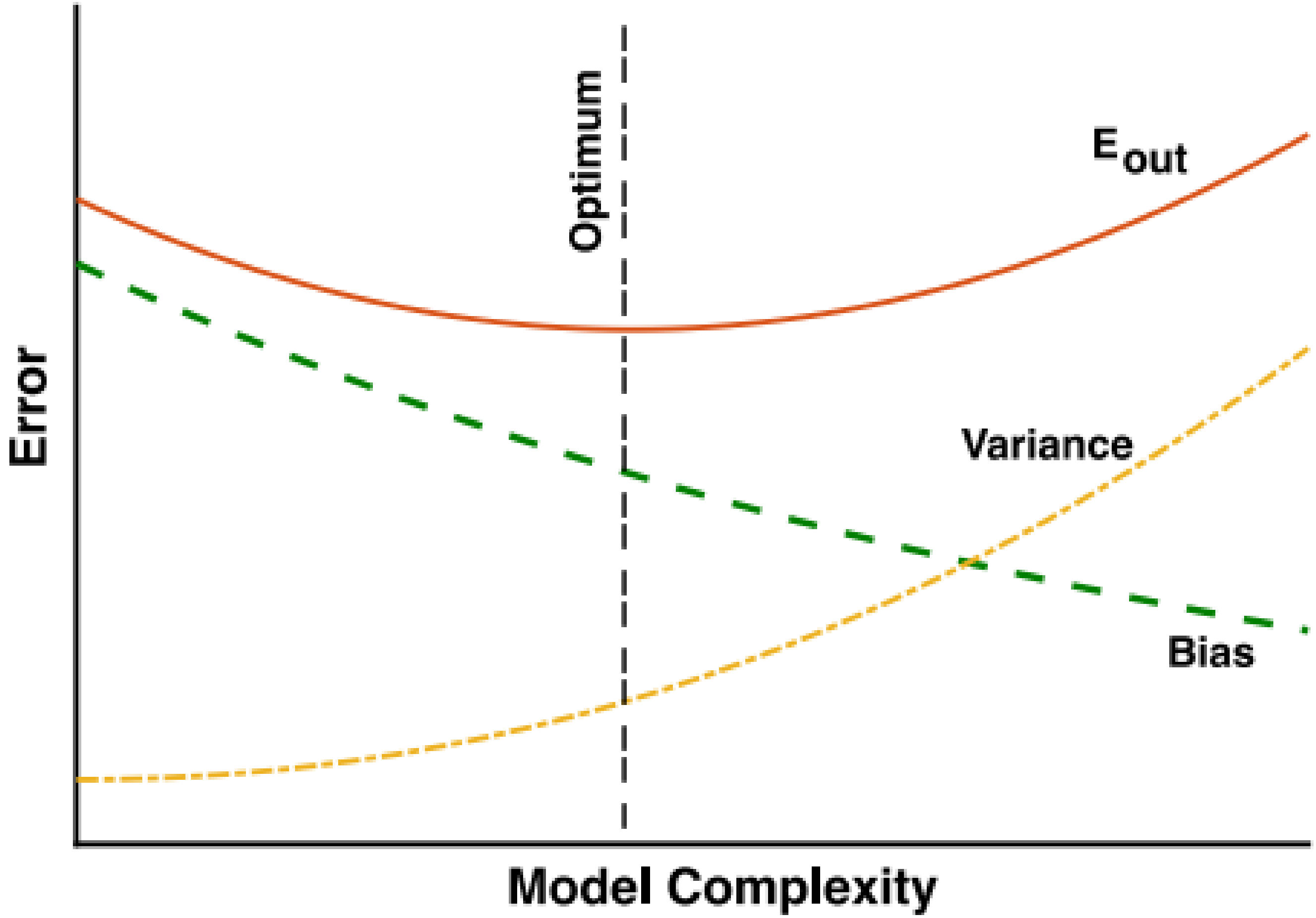


- Enough data lowers variance
- However, we are always working in the low data regime

Mehta et al. Phys. Rep. **810** (2019)

Bias variance tradeoff

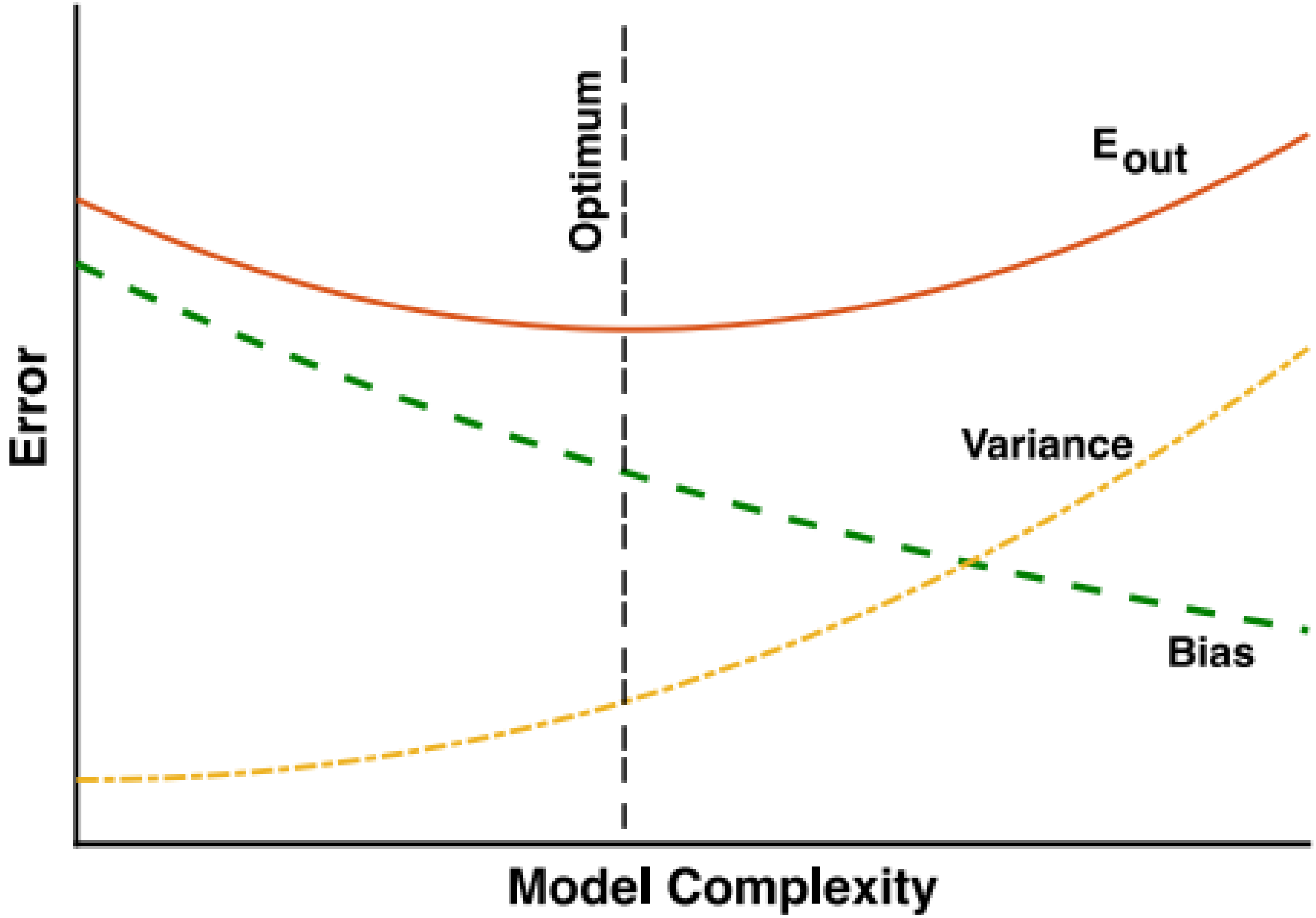
- There is an optimum size of your model for a given amount of data.



Mehta et al. Phys. Rep. **810** (2019)

Bias variance tradeoff

- There is an optimum size of your model for a given amount of data.



- L2 regularization (also known as weight decay):
  - lowers variance/prevents overfitting
  - Allows you to use larger models while getting lower errors

Use the right amount of regularization

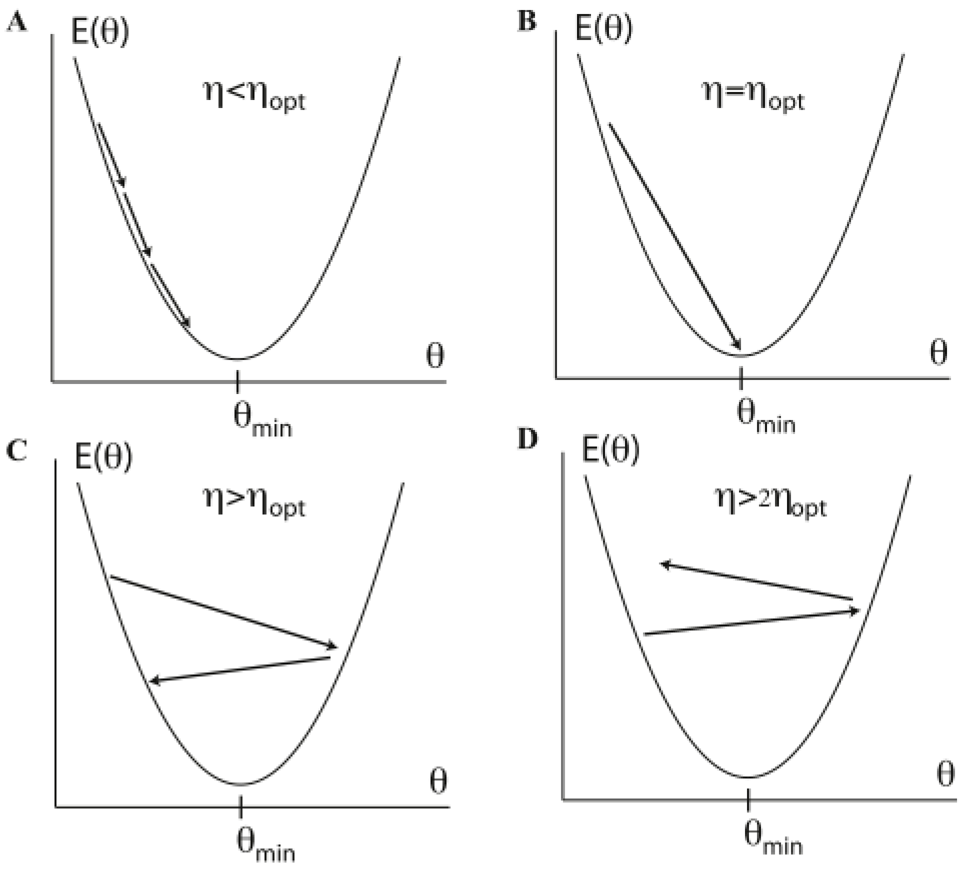
Mehta et al. Phys. Rep. **810** (2019)

Find $\min_{\boldsymbol{\theta}} \mathrm{C}\big(\mathbf{y}, \mathrm{f}(\mathbf{X}; \boldsymbol{\theta})\big)$: Gradient descent

$$\boldsymbol{v}_t = -\eta \nabla_{\boldsymbol{\theta}} \mathrm{C}\big(\mathbf{y}, \mathrm{f}(\mathbf{X}; \boldsymbol{\theta})\big)$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{v}_t$$

η step size
v_t update to weights

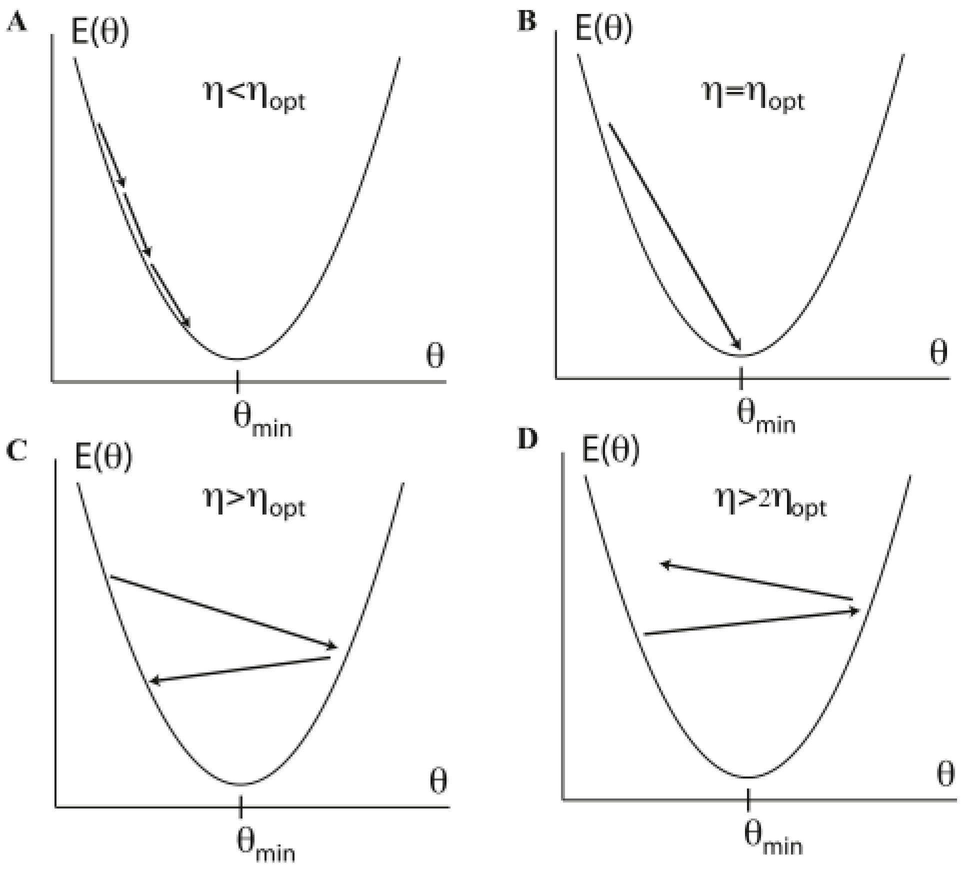


Mehta et al. Phys. Rep. **810** (2019)

Find $\min_{\boldsymbol{\theta}} \mathrm{C}\big(\mathbf{y}, \mathrm{f}(\mathbf{X}; \boldsymbol{\theta})\big)$: Gradient descent

$$\boldsymbol{v}_t = -\eta \nabla_{\boldsymbol{\theta}} \mathrm{C}\big(\mathbf{y}, \mathrm{f}(\mathbf{X}; \boldsymbol{\theta})\big)$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{v}_t$$



- Momentum algorithms
  - Build up speed in shallow directions

$$\boldsymbol{v}_t = \gamma \boldsymbol{v}_{t-1} - \eta \nabla_{\boldsymbol{\theta}} \mathrm{C}\big(\mathbf{y}, \mathrm{f}(\mathbf{X}; \boldsymbol{\theta})\big)$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{v}_t$$

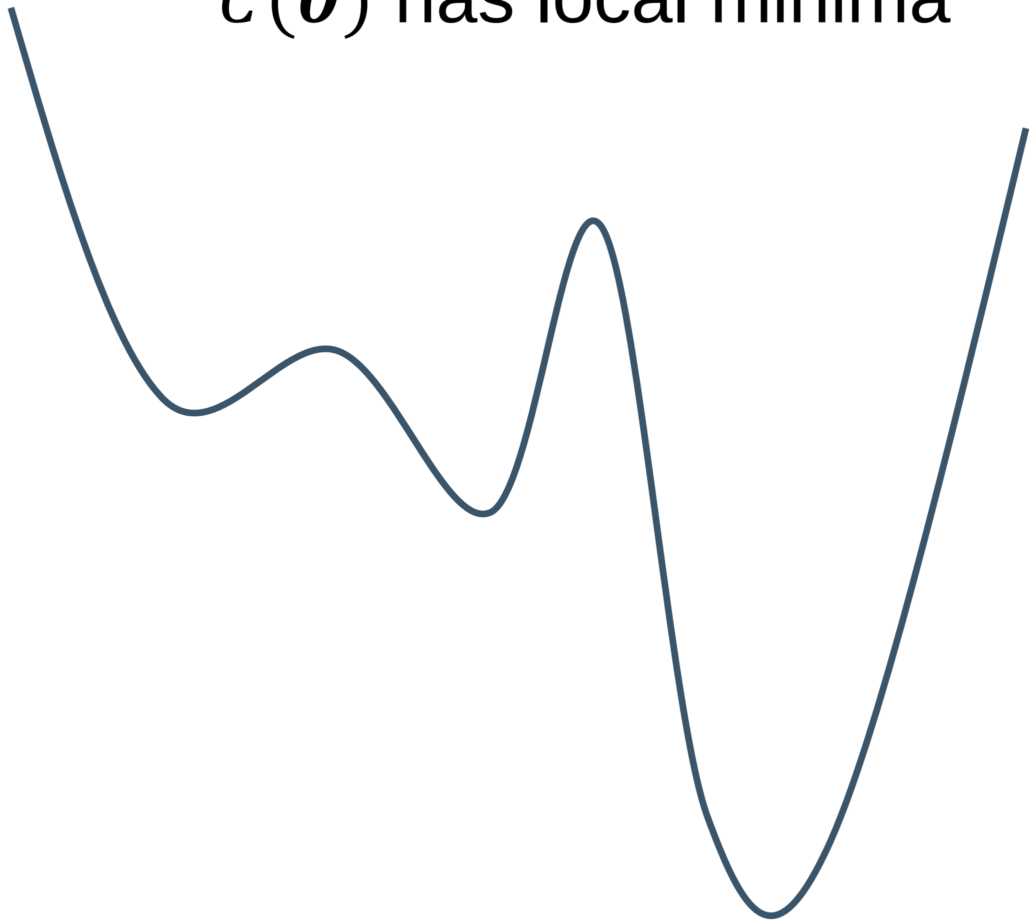
Mehta et al. Phys. Rep. **810** (2019)

Momentum

$$\boldsymbol{v}_t = -\eta \nabla_{\boldsymbol{\theta}} \mathrm{C}\big(\mathbf{y}, \mathrm{f}(\mathbf{X}; \boldsymbol{\theta})\big)$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{v}_t$$

$C(\boldsymbol{\theta})$ has local minima



- Momentum algorithms
  - Build up speed in shallow directions
  - Can get out of local minima

$$\boldsymbol{v}_t = \gamma \boldsymbol{v}_{t-1} - \eta \nabla_{\boldsymbol{\theta}} \mathrm{C}\big(\mathbf{y}, \mathrm{f}(\mathbf{X}; \boldsymbol{\theta})\big)$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{v}_t$$

Mehta et al. Phys. Rep. **810** (2019)

Use stochasticity to get out of local minima

$$\boldsymbol{v}_t = -\eta \nabla_{\boldsymbol{\theta}} \mathrm{C}\big(\mathbf{y}, \mathrm{f}(\mathbf{X}; \boldsymbol{\theta})\big)$$

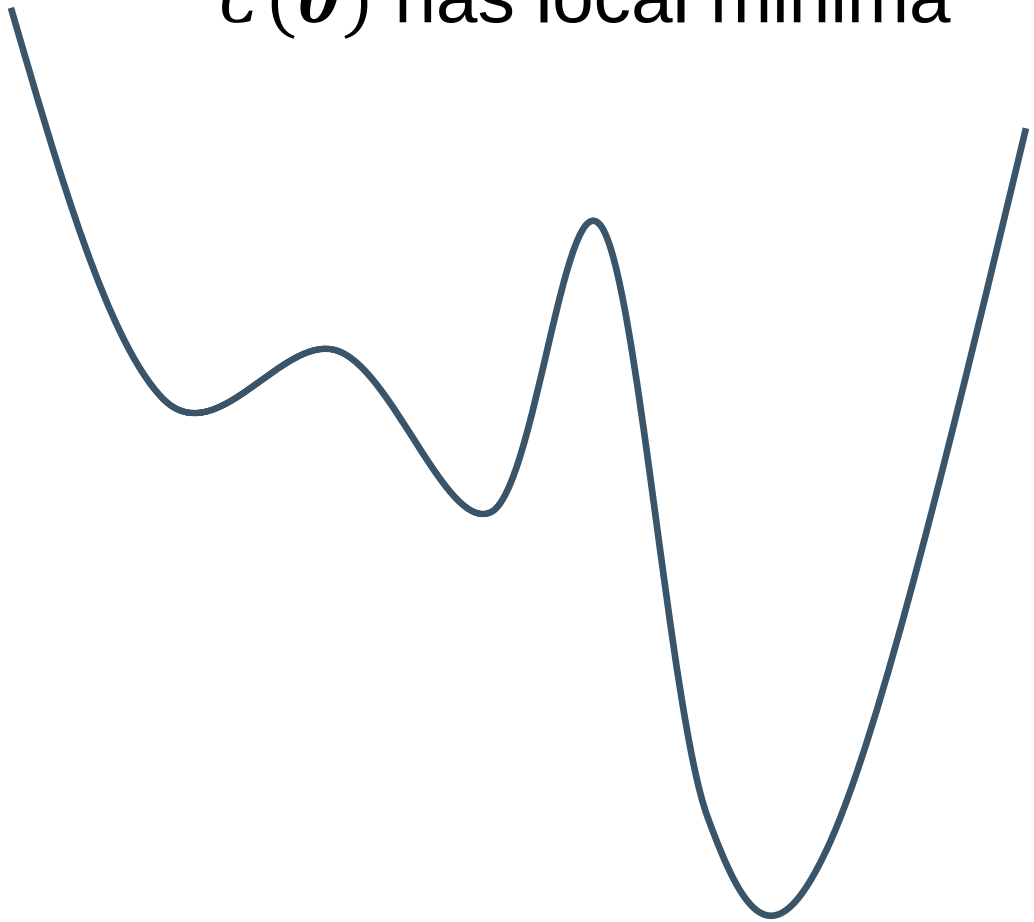
$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{v}_t$$

Only compute v_t on a subset of X and y

$$\boldsymbol{v}_t = -\eta \nabla_{\boldsymbol{\theta}} \mathrm{C}\big(\mathbf{y}_{\mathrm{batch}}, \mathrm{f}(\mathbf{X}_{\mathrm{batch}}; \boldsymbol{\theta})\big)$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \boldsymbol{v}_t$$

$C(\boldsymbol{\theta})$ has local minima



Mehta et al. Phys. Rep. **810** (2019)

A model to play with

- Run lj_3_hyperparameters.py

- Change hyperparameters at the top of the script and see how the training progression changes

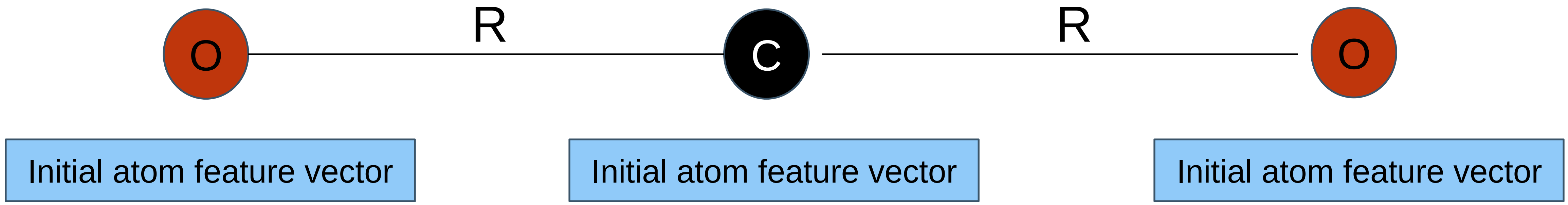
- Tensorboard to plot training progression
  - Using the anaconda prompt, in the ml_tutorial folder run:

  tensorboard --logdir=runs --reload_multifile True
  - Go to http://localhost:6006/ in your browser

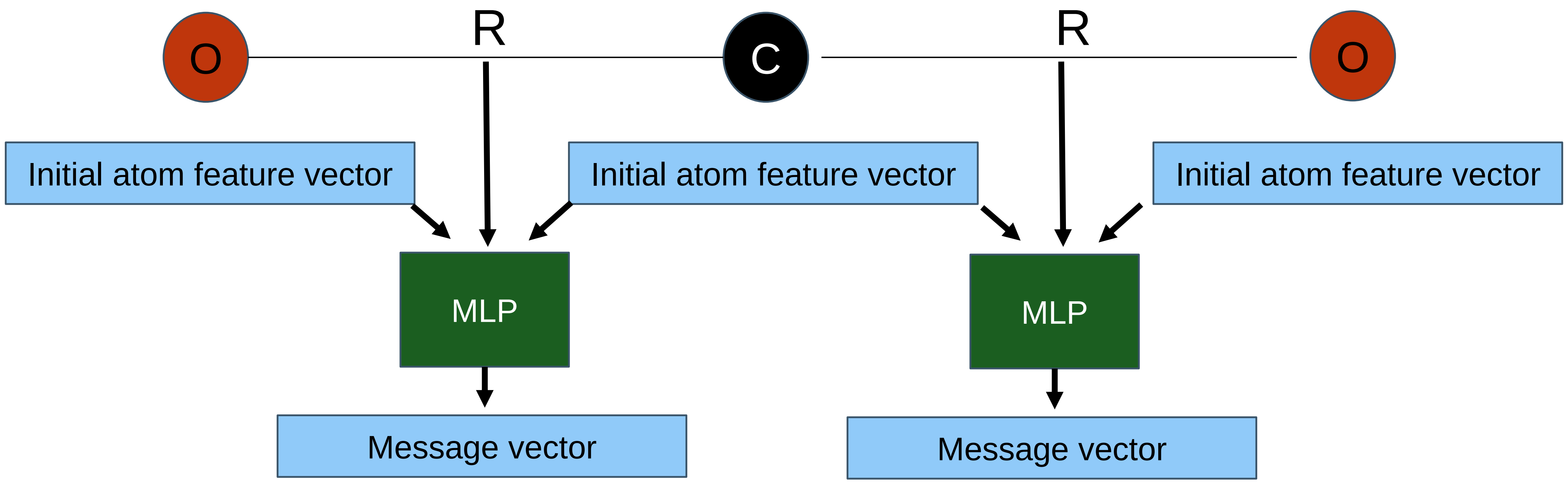
Best practices when developing an NN

- Try to memorize a few data points first
- Try also using fake simple data
- Check your descriptors and targets to make sure you are feeding the NN what you think you are
- Hyperparameter tuning on even just a few epochs to screen out unpromising parameter values
- Use grid search/random search of hyperparameters

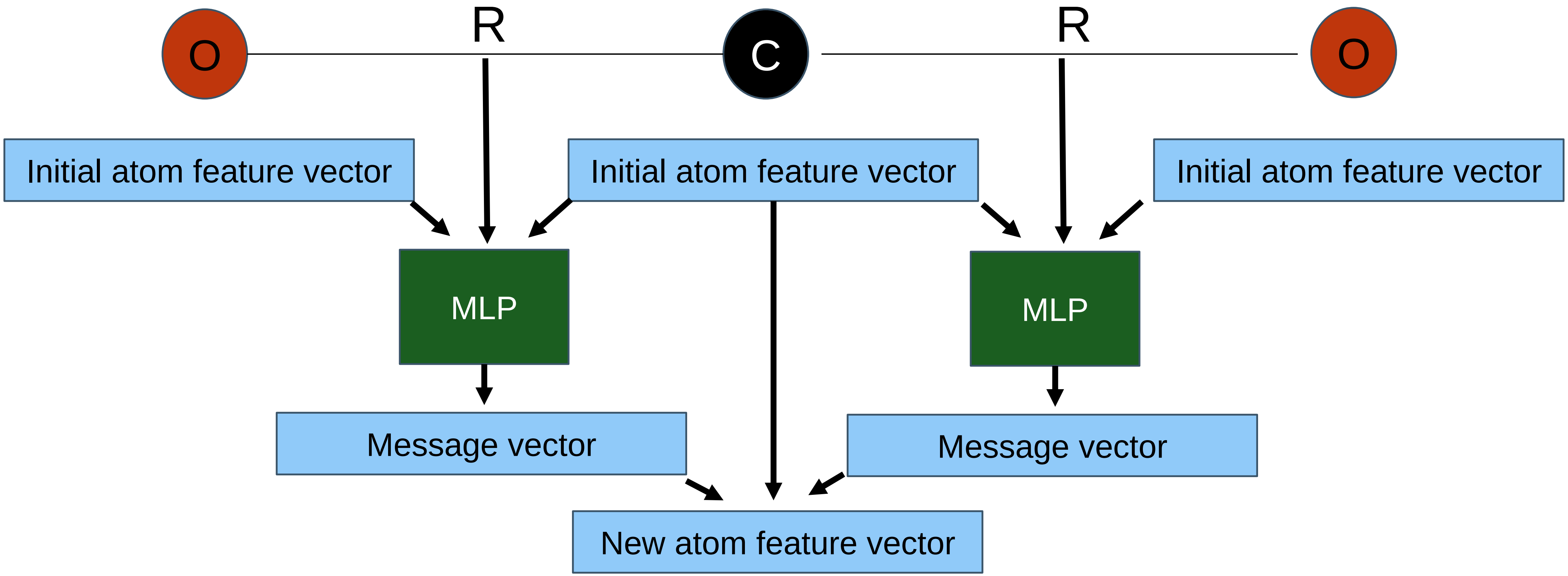
Architecture of message passing neural networks



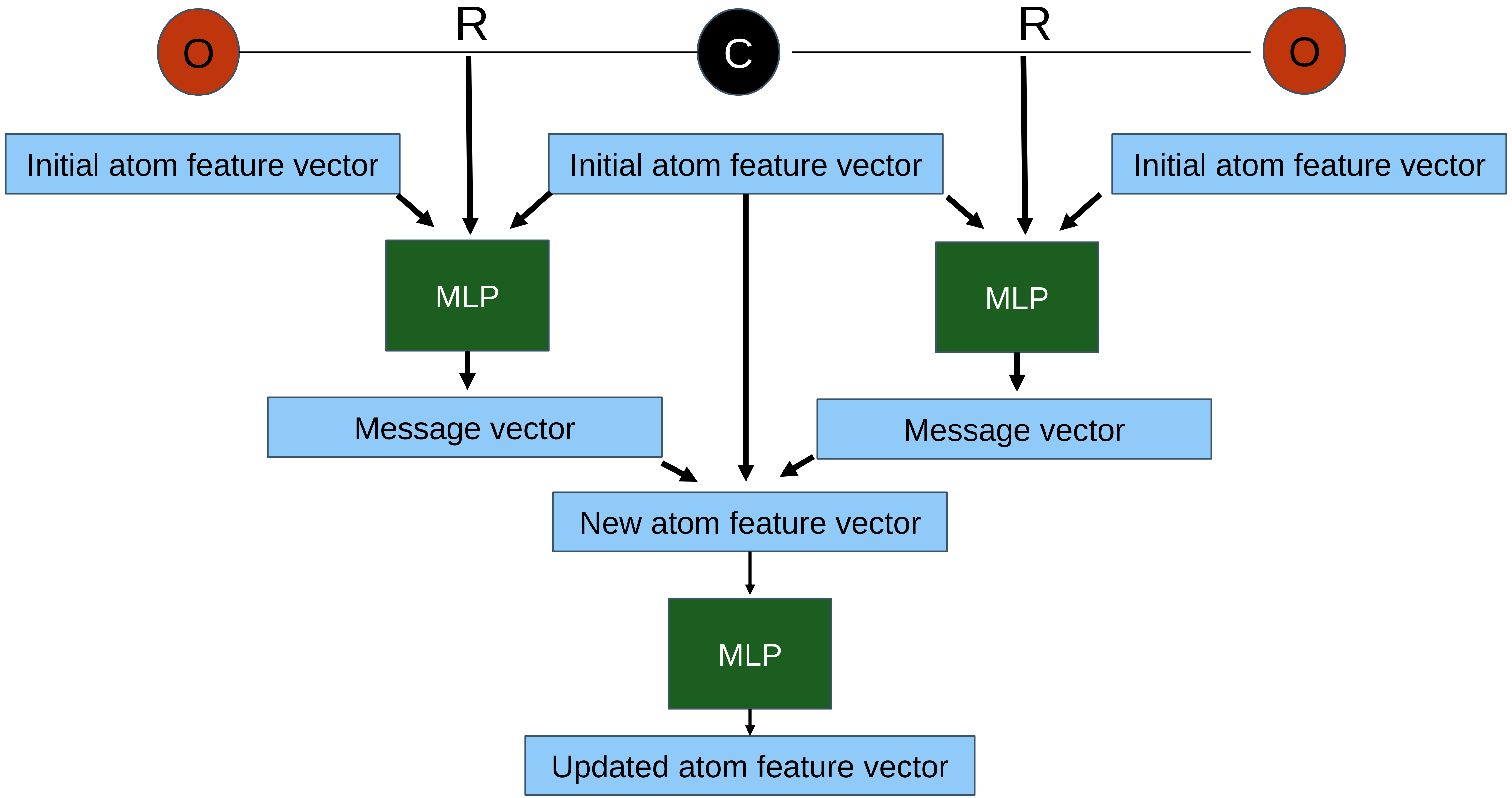
Architecture of message passing neural networks



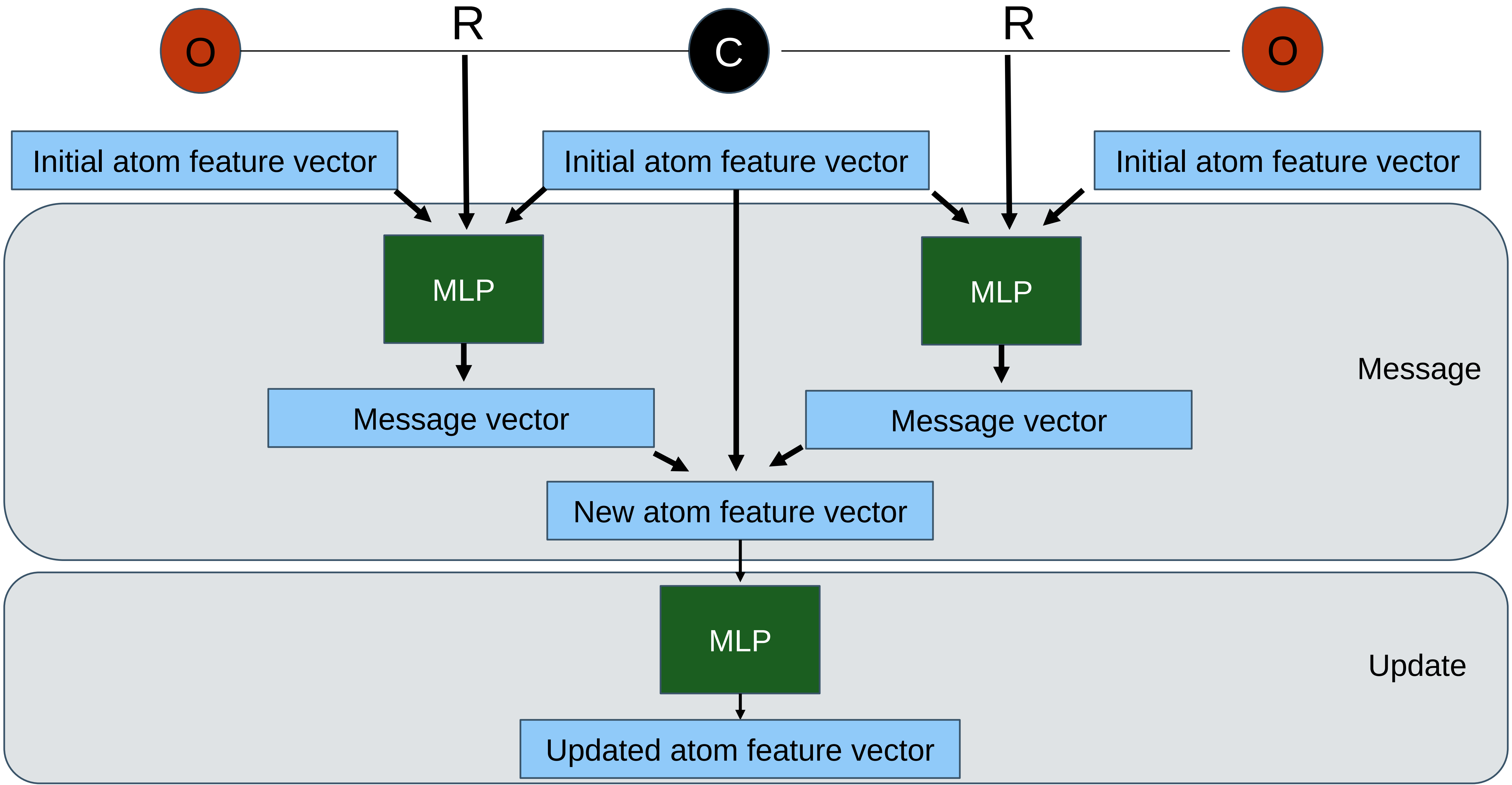
Architecture of message passing neural networks



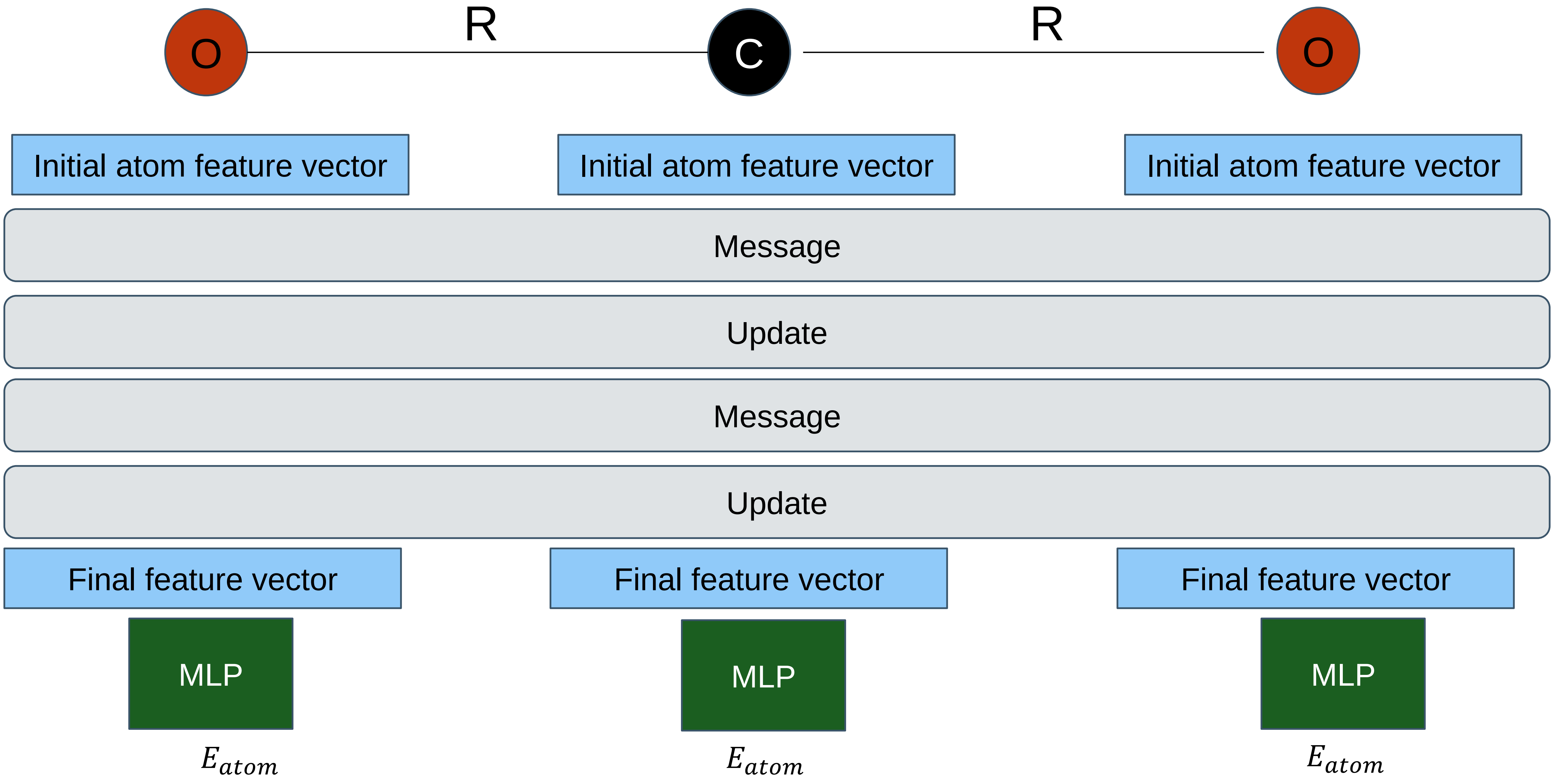
Architecture of message passing neural networks



Architecture of message passing neural networks



Architecture of message passing neural networks



Acknowledgements

Theoretical Chemical Physics Group
University of Luxembourg

Prof. Alexandre Tkatechnko

Artem Kokorin

Funded by the European Union