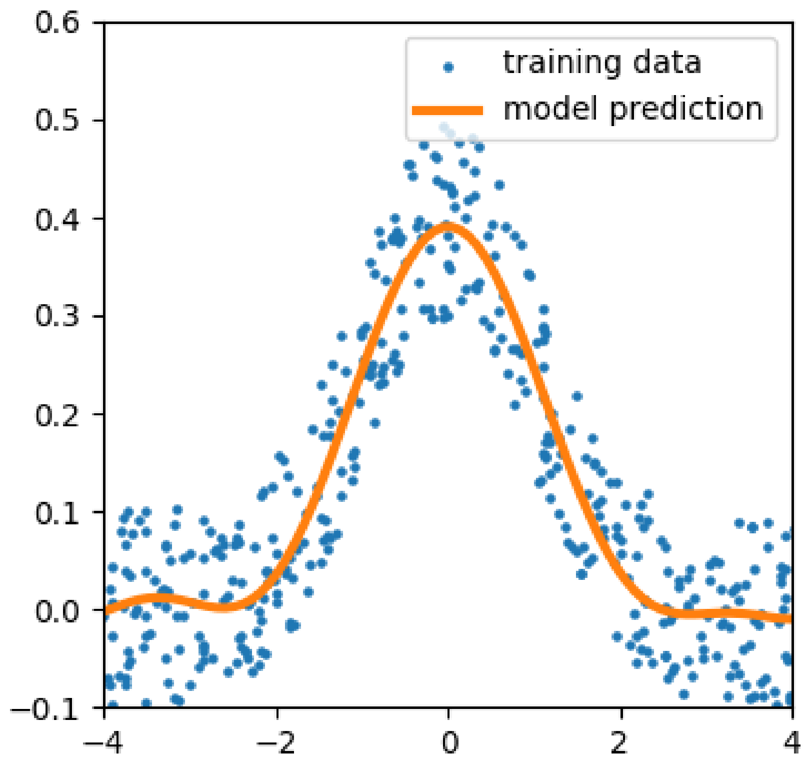
A jumpstart into machine-learning

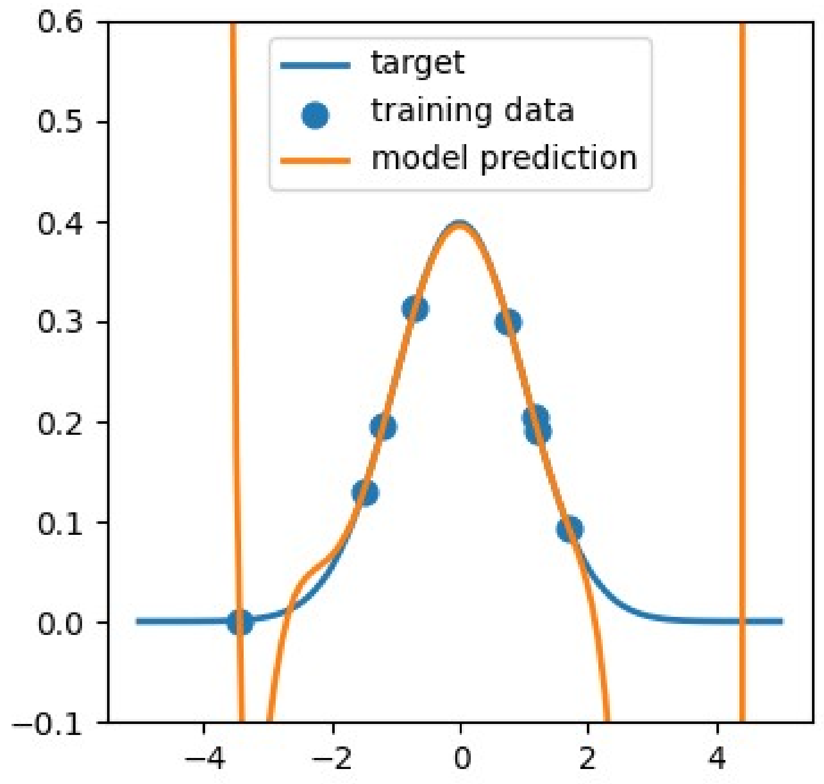
Dahvyd Wing

Regression for computational chemistry

Most applications: lots of noisy data



Our case: few data points with almost no noise



Anatomy of regression

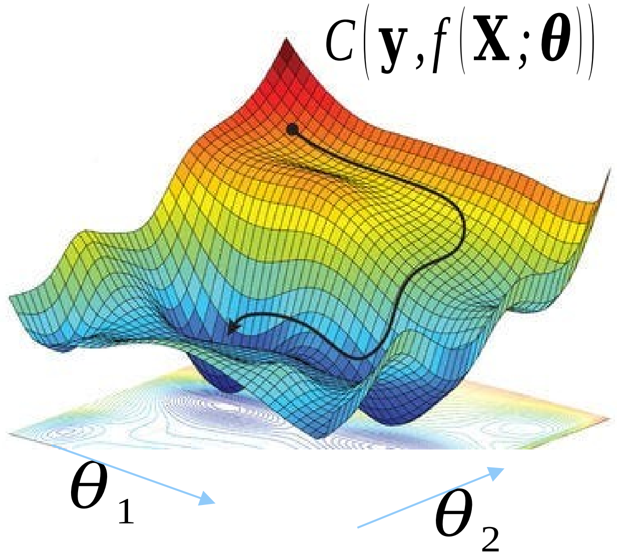
1. Data:

2. Model:

   are trainable parameters

3. Cost function:
   mean squared error (MSE)

4. Find 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:
   - Instance/object of a customized dataset class
   - Implement 3 functions: _init_, _len_, and _get_item_
   - dataloader pulls random batches of data from the dataset

$$\mathbf{x} = ¿$$

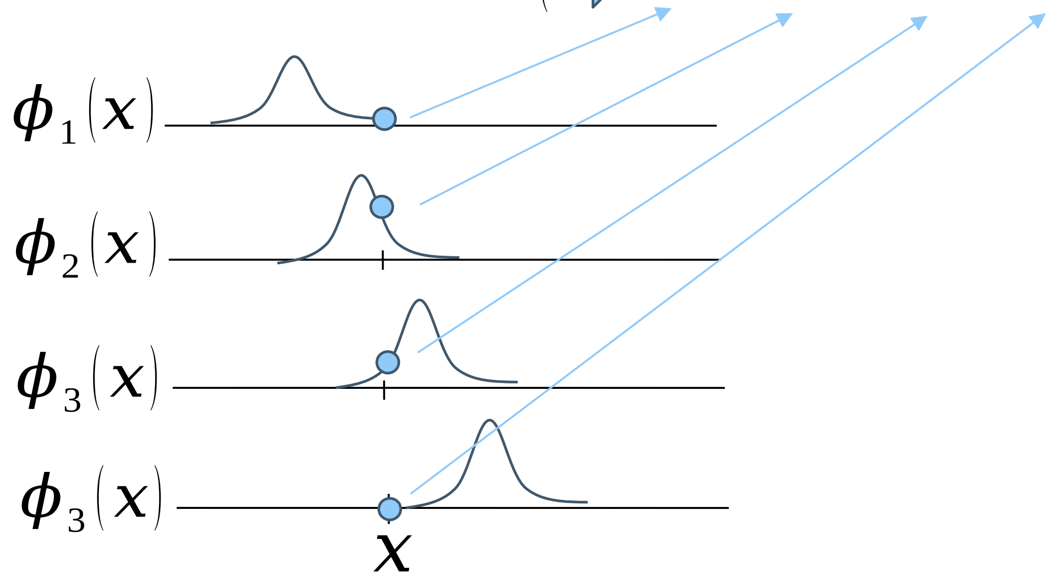
Anatomy of regression

1. Data:

2. Model:
   - Instance/object of a customized nn.module class
   - Implement 2 functions: _init_ and _forward_
   - Descriptor:

One hot encoding:

Continuous data: $x = 3.1 \Rightarrow \begin{pmatrix} 0.1 & 0.5 & 0.3 & 0.01 & 0 & 0 \end{pmatrix}$

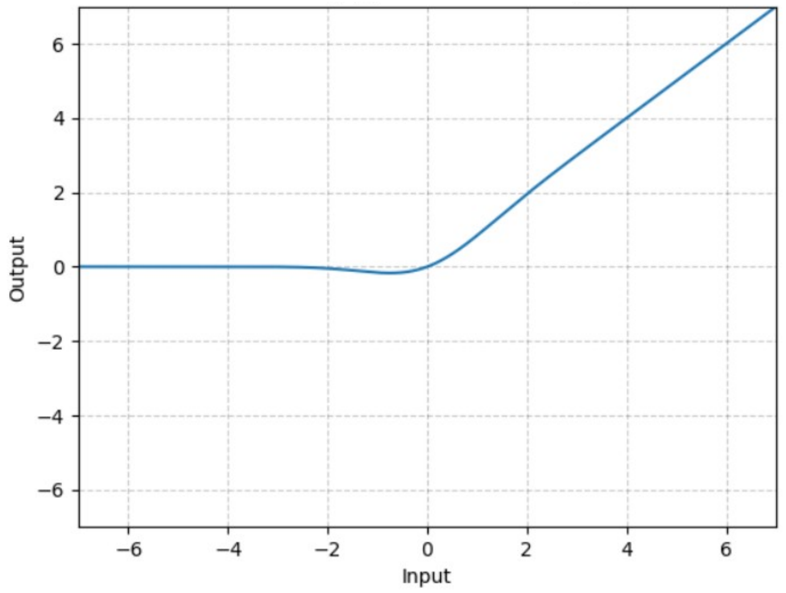


Anatomy of regression

1. Data:

2. Model:
   - Instance/object of a customized nn.module class
   - Implement 2 functions: _init_ and _forward_
   - Descriptor
   - Neural network:

is the nonlinear activation function: GELU



Use a continuously differentiable activation function

$$y_{pred} = \mathbf{w_3} \cdot \mathbf{y_3} + b_3$$

Anatomy of regression

1. Data:

2. Model:

3. Cost function:
   - Mean squared error

Anatomy of regression

1. Data:
2. Model:
3. Cost function:
4. Find

Batch 1

, ,

Batch 2

, ,

Batch 3

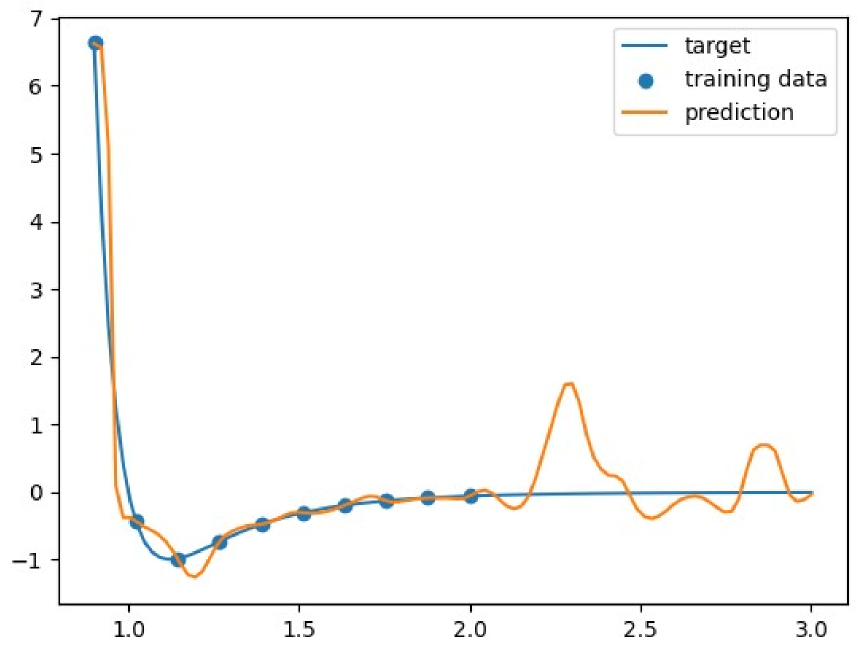
, ,

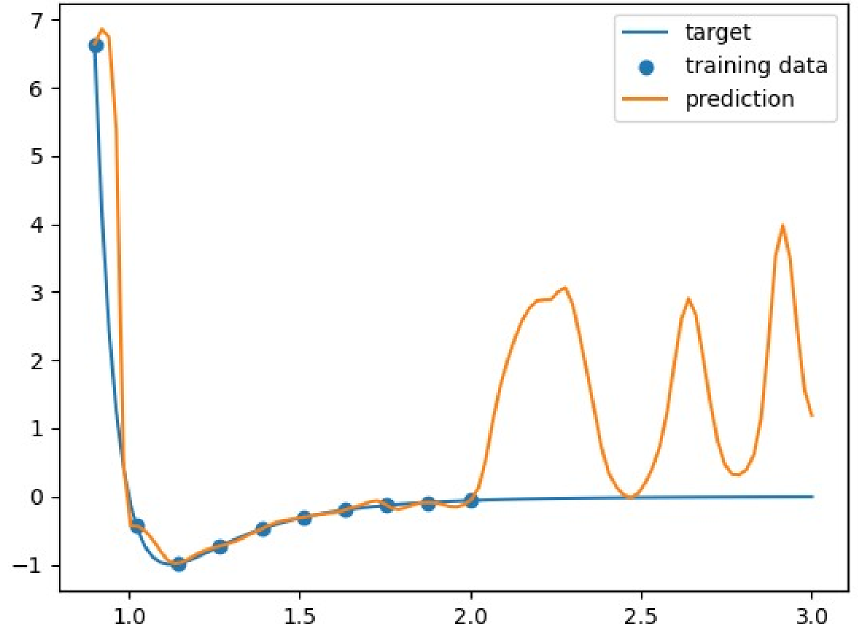
…

$$\underbrace{\boldsymbol{\theta}' = \boldsymbol{\theta} - \nabla_{\boldsymbol{\theta}} C\left(\mathbf{y}, f\left(\mathbf{X}; \boldsymbol{\theta}\right)\right) \qquad \boldsymbol{\theta}' = \boldsymbol{\theta} - \nabla_{\boldsymbol{\theta}} C\left(\mathbf{y}, f\left(\mathbf{X}; \boldsymbol{\theta}\right)\right) \qquad \boldsymbol{\theta}' = \boldsymbol{\theta} - \nabla_{\boldsymbol{\theta}} C\left(\mathbf{y}, f\left(\mathbf{X}; \boldsymbol{\theta}\right)\right)}_{\text{1 Epoch}}$$

Overfitting

- NNs often have many more parameters than samples in the training data
- Run lj_1_overfit.py several times
  - 6,601 trainable parameters, 10 data points



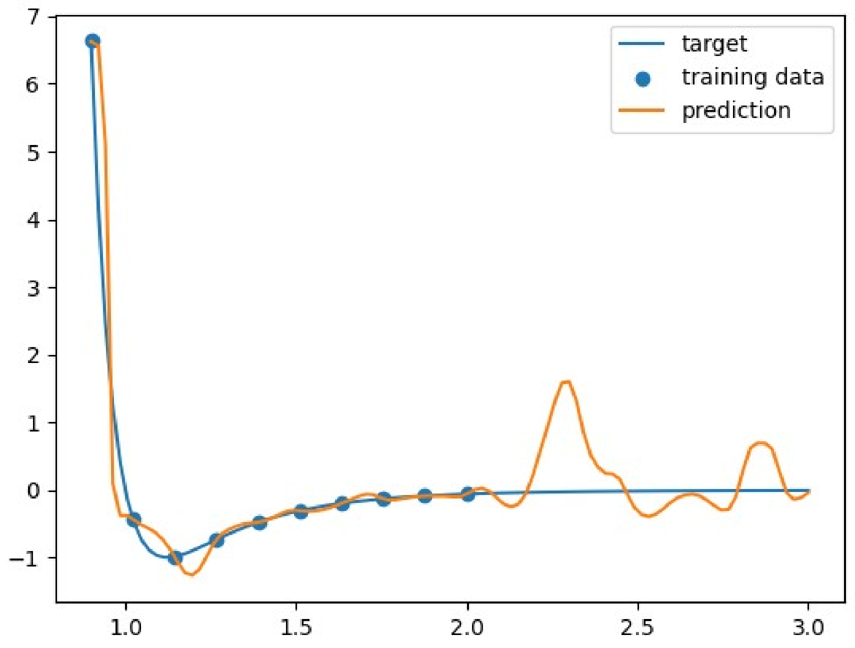


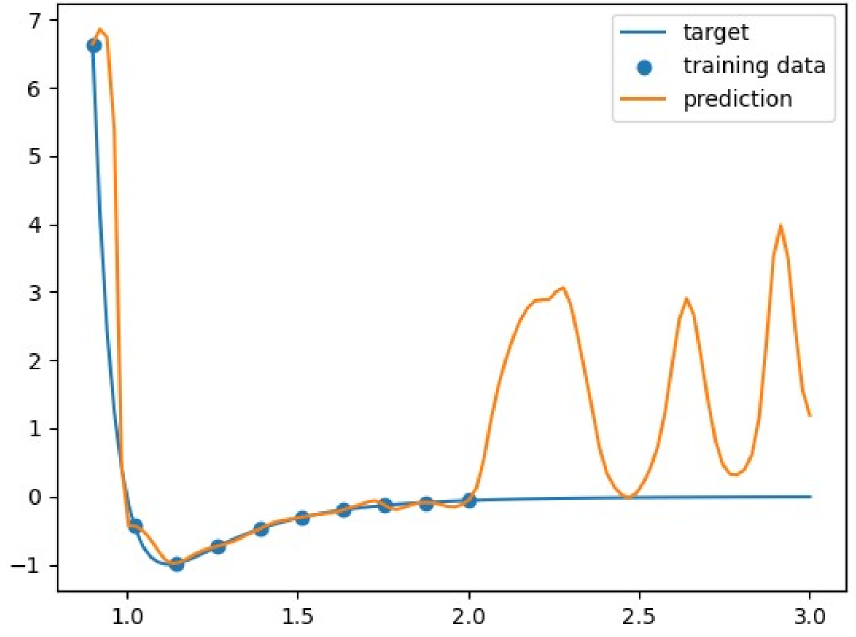
- Each model perfectly fits the training points, but doesn’t do a great job in the interpolation region

You measure a model by testing on data it has never seen

Overfitting

- NNs often have many more parameters than samples in the training data
- Run lj_1_overfit.py several times
  - 6,601 trainable parameters, 10 data points





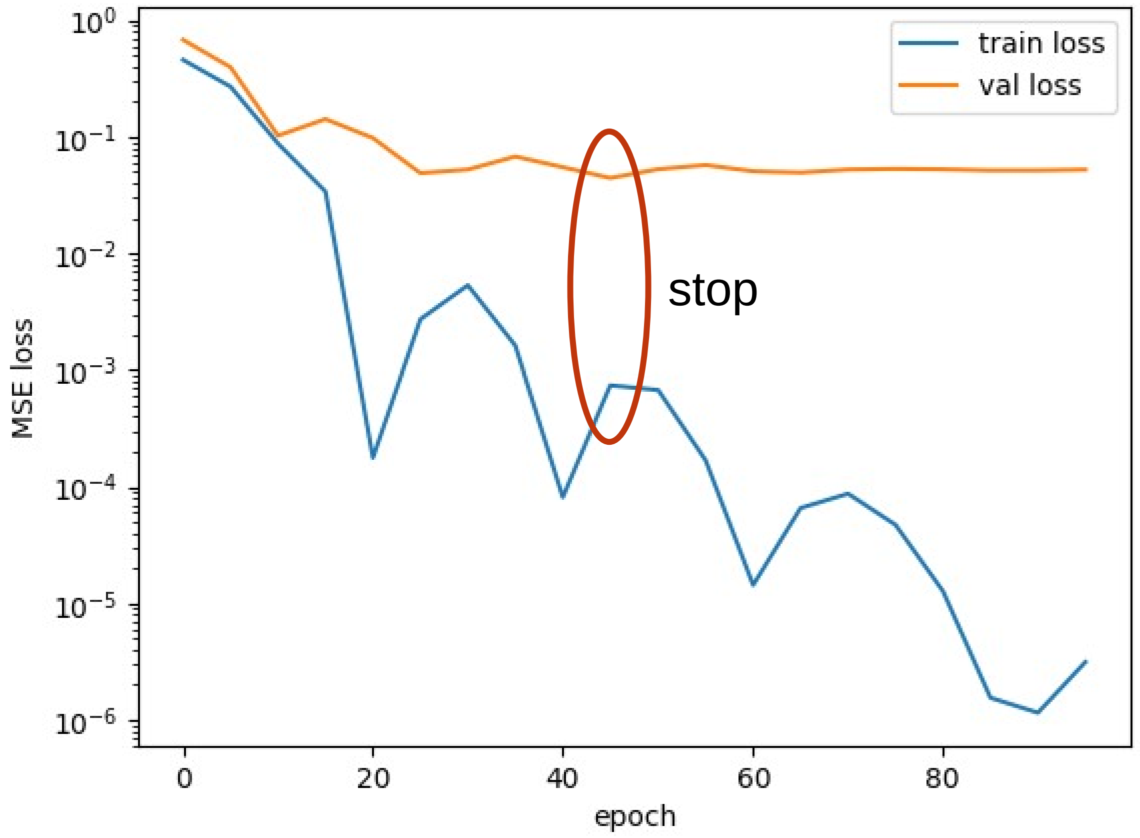
- Each model perfectly fits the training points, but doesn’t do a great job in the interpolation region
- The models do terrible in the extrapolation regime

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

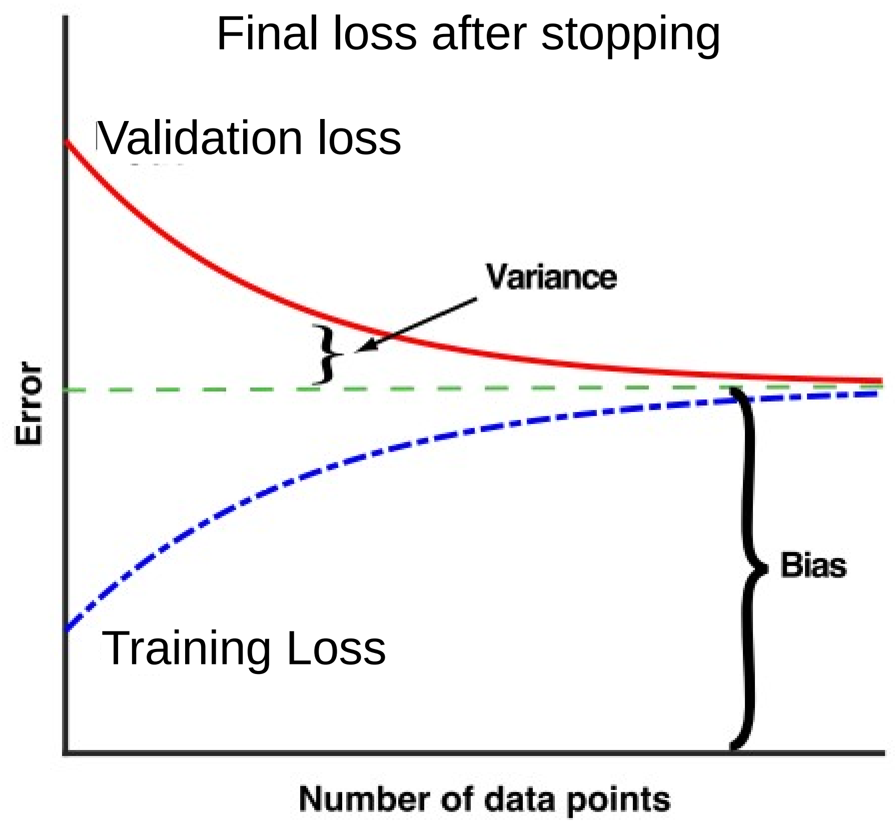
- Run lj_2_overfit_with_validation.py
  - The main change in the code is lines 56-58

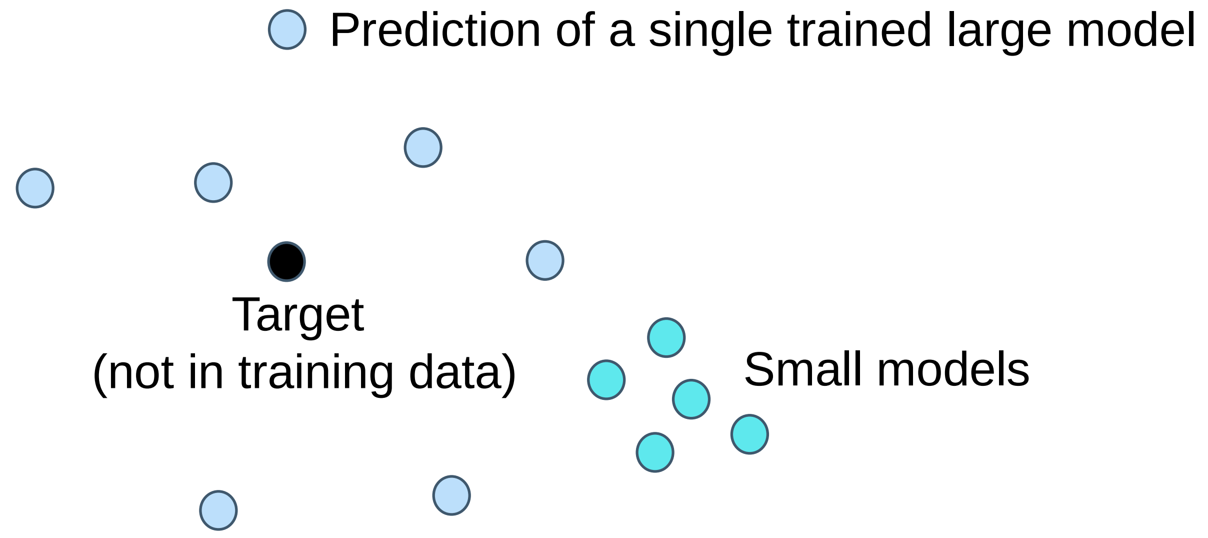
- To get the best performance model stop when there is a steady increase in validation loss and decrease in training loss (early stopping)



Learning curve

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

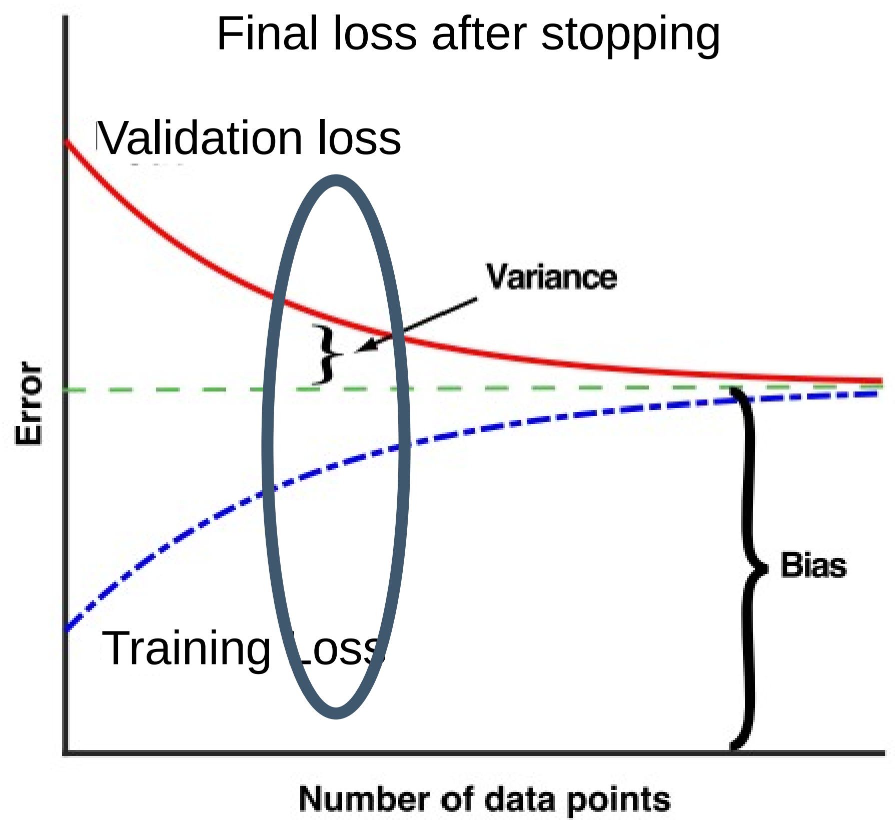




Mehta et al. Phys. Rep. (2019)

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

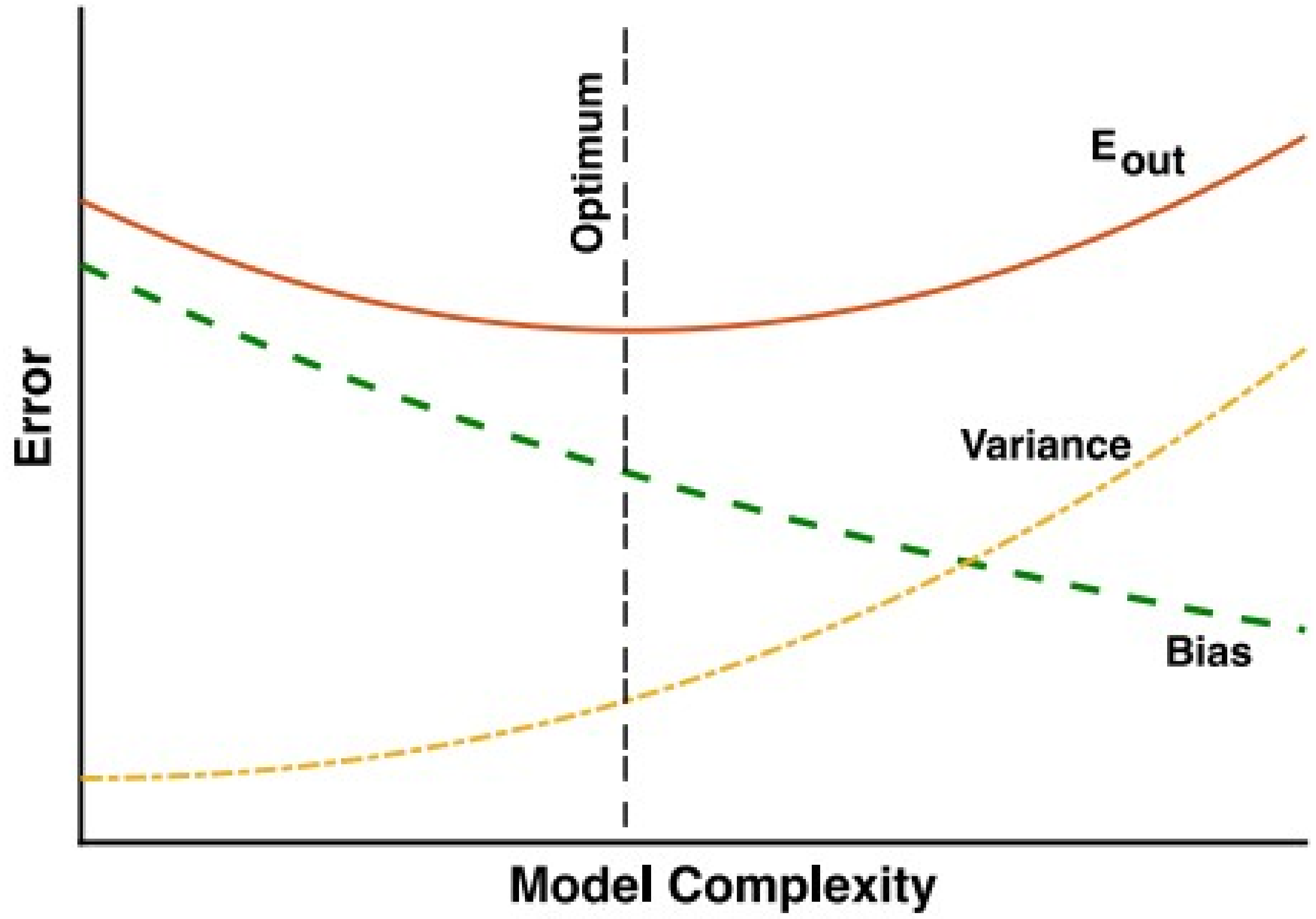


- Enough data removes variance
- We are always working in the low data regime

Mehta et al. Phys. Rep. (2019)

Bias variance tradeoff

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



- L2 regularization lowers variance.
  - Prevents overfitting
  - Allows you to use larger models

Use the right amount of regularization

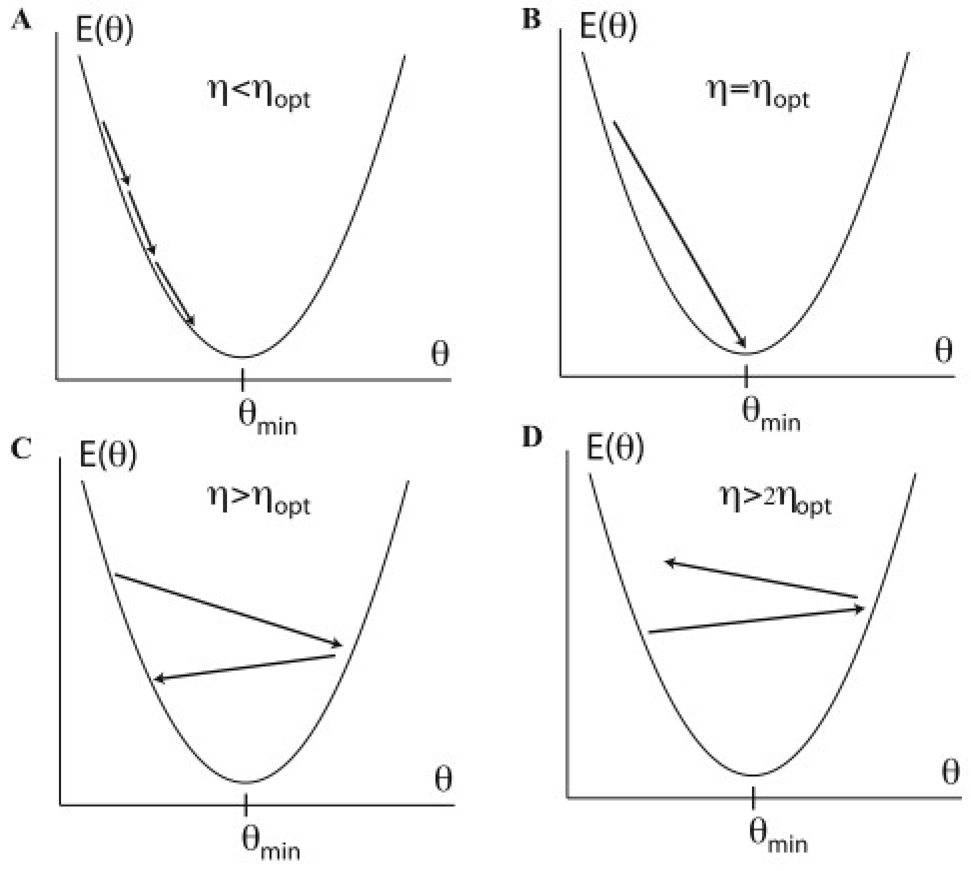
Mehta et al. Phys. Rep. (2019)

Find : Gradient descent

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

- Momentum algorithms
  - Build up speed in shallow directions

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



Mehta et al. Phys. Rep. (2019)

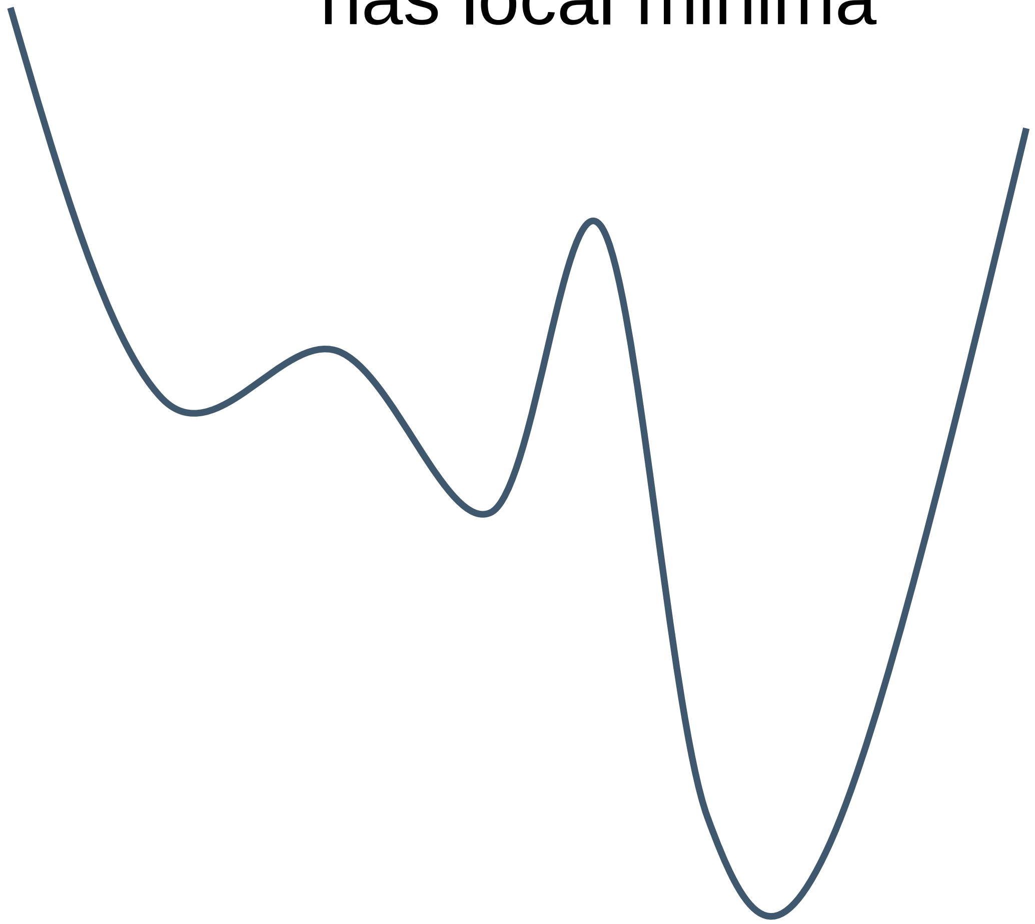
Momentum

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

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

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

has local minima



Mehta et al. Phys. Rep. (2019)

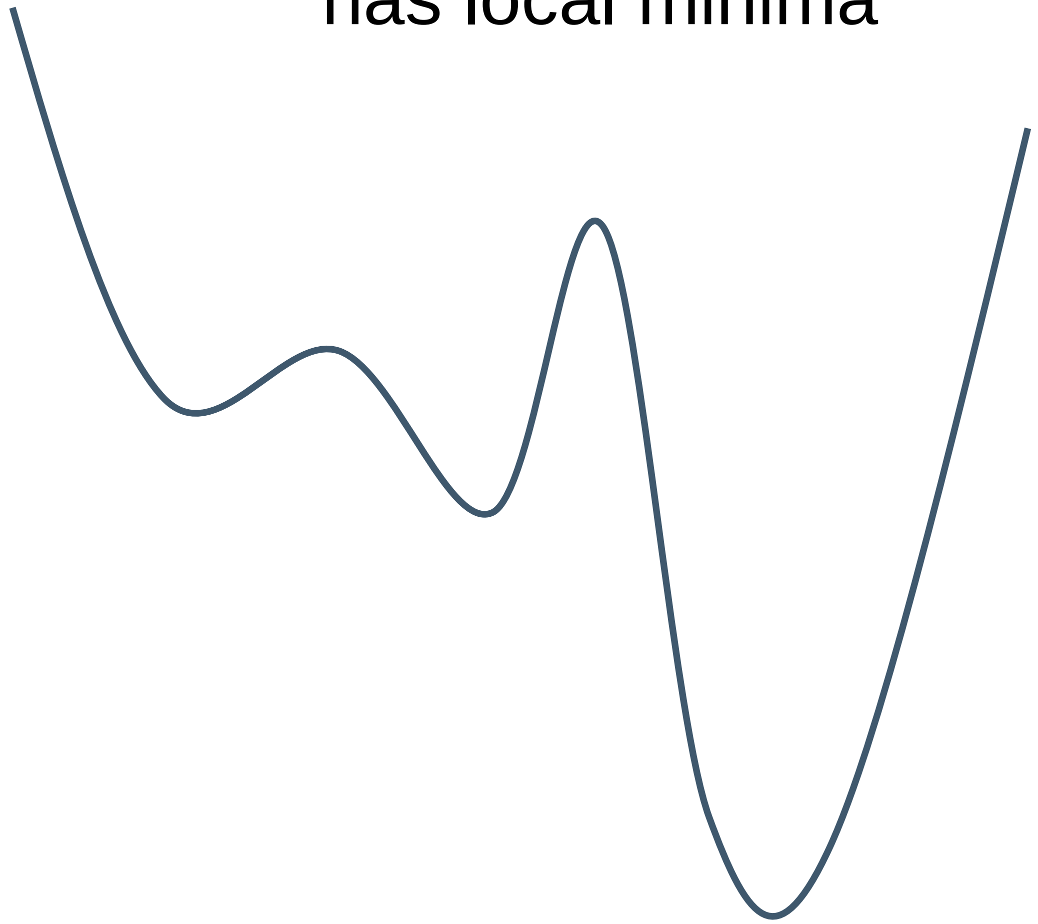
Use stochasticity to get out of local minima

has local minima

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

Only compute on a subset of and y

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



Mehta et al. Phys. Rep. (2019)

A model to play with

- Run lj_3_hyperparameters.py

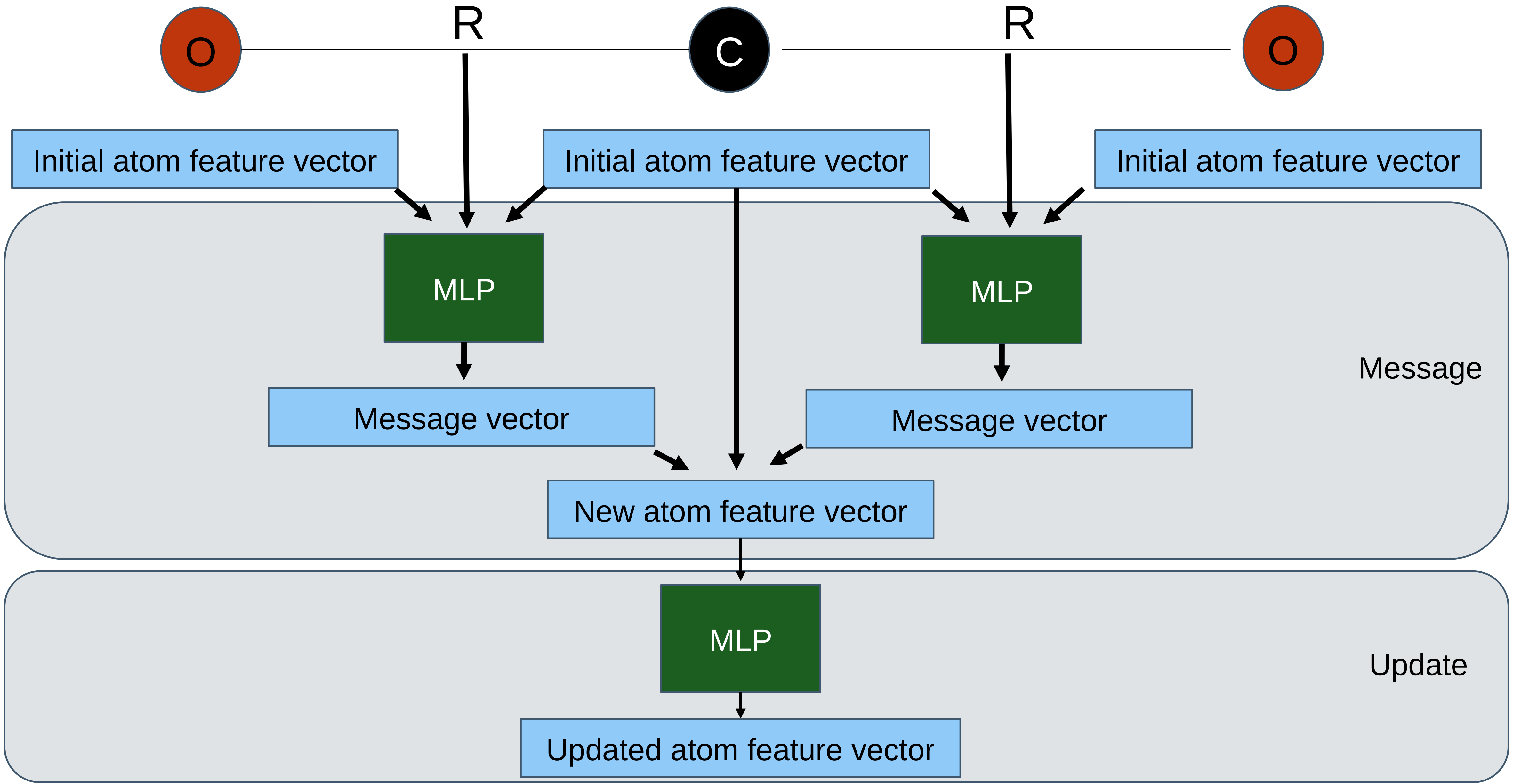
- Change hyperparameters at the top of the script and see how the training progression changes

- Tensorboard to plot training progression
  - In ml_tutorial folder run: tensorboard --logdir=runs --reload_multifile True
  - Go to http://localhost:6006/ in your browser

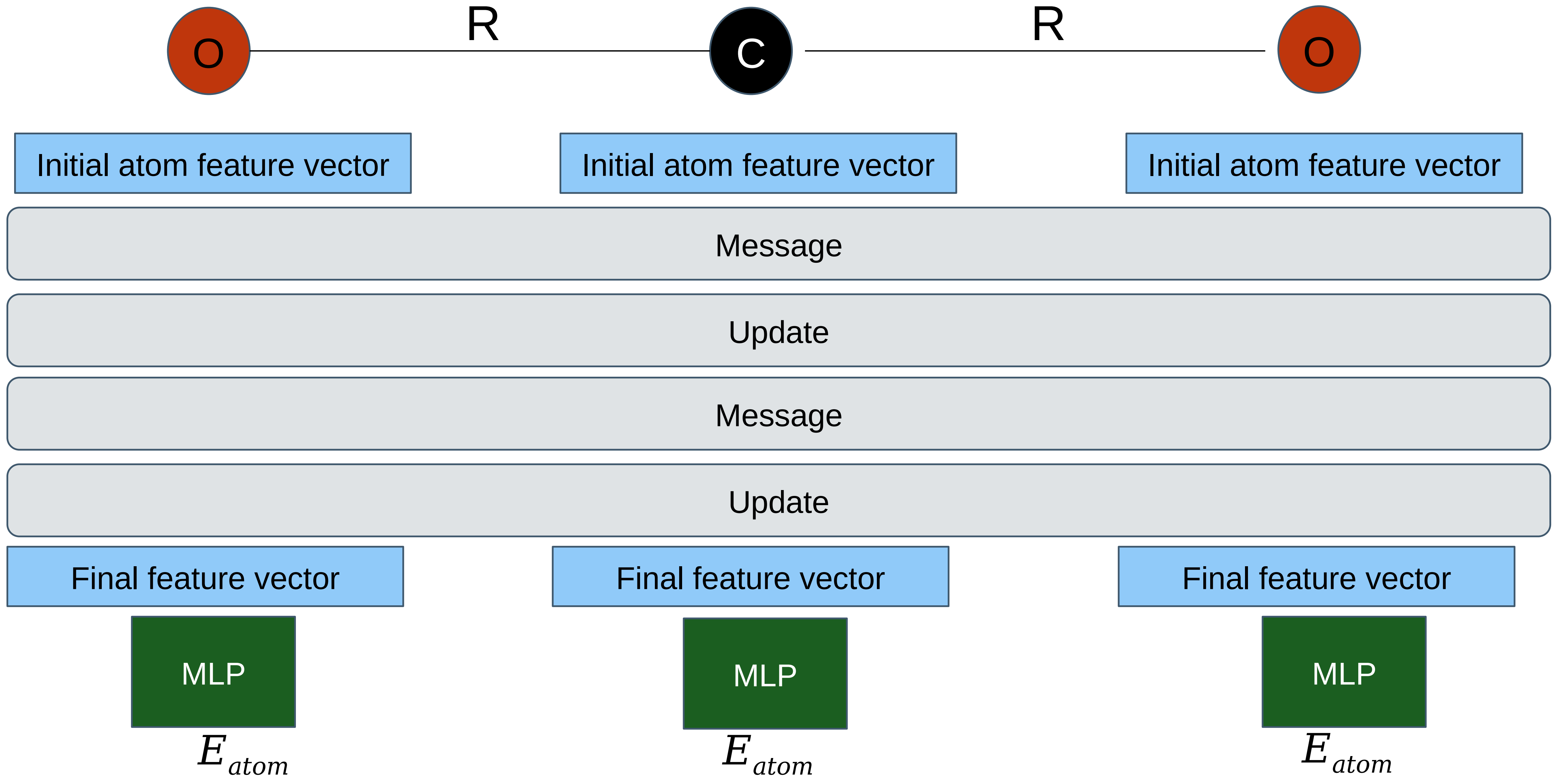
Best practices when developing an NN

- 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


O
R
C
R
O
Initial atom feature vector
Initial atom feature vector
Initial atom feature vector
MLP
MLP
Message vector
Message vector
Message
New atom feature vector
MLP
Update
Updated atom feature vector

Architecture of message passing neural networks



Getting to molecular dynamics in 1 minute

- In schnet_tutorial folder run script2_train_schnet.py
  - Trains a schnet model on 1000 data points of a ethanol DFT(PBE) MD trajectory for 5 epochs
- run script3_run_md.py
  - Creates an atomic structure environment (ASE) force field calculator using the schnet model. Uses ASE to run an MD trajectory
  - Note: that you can only run this script once, it will throw an error if you run it a second time

- in anaconda prompt run: ase gui
- file -> open -> ase_calcs (in left pane) -> simulation.traj
- tools -> movies -> play

- Note that the state-of-the-art ML-FF's are equivariant (PAINN, NequIP, etc.). Schnet is an invariant model that's cheap to train and run.

Schutt et al. J. Chem. Phys. 158 (2023)

Acknowledgements

Prof. Alexandre Tkatechnko

Artem Kokorin

Funded by the European Union

Thank You!