Training Models

Using Machine Learning Tools

Geron Chapter 4

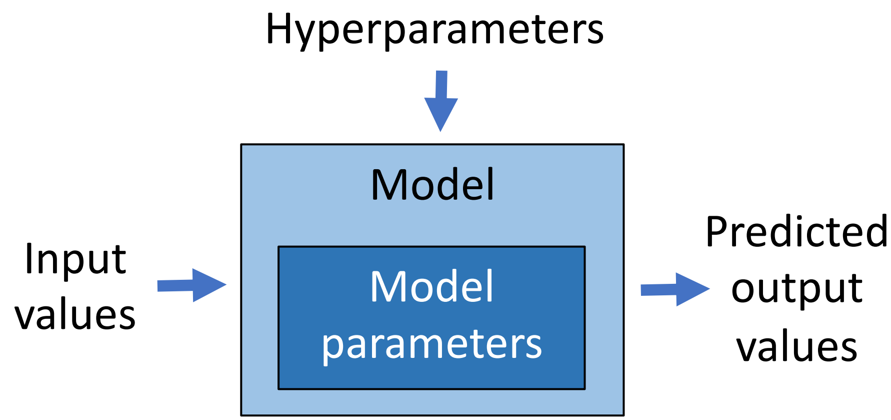
Last Time …

- Classification vs. regression
- Range of classifier approaches, white box vs. black box
- Several performance metrics
  - Accuracy, Confusion matrix
  - Precision, Recall
  - ROC, AUC
- Look at the data at all stages:
  - scatter plot
  - model parameters / decision boundaries

- Today we will look at training (i.e. optimisation of model parameters) in more detail

Model

- A mapping of input values to predicted outcome values
- Flexible due to model parameters
- Constrained by fixed hyper-parameters



- Example: Linear model

Model parameters

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = \boldsymbol{\theta}^T \cdot \boldsymbol{x} = h_{\boldsymbol{\theta}}(\boldsymbol{x})$$

Predicted output value

Input values for ***one sample***

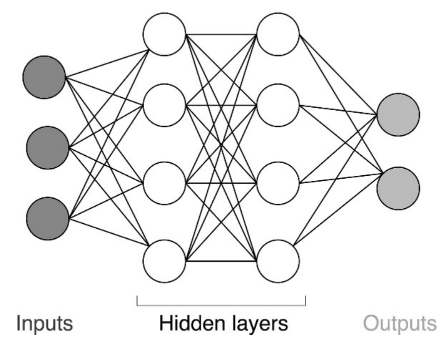
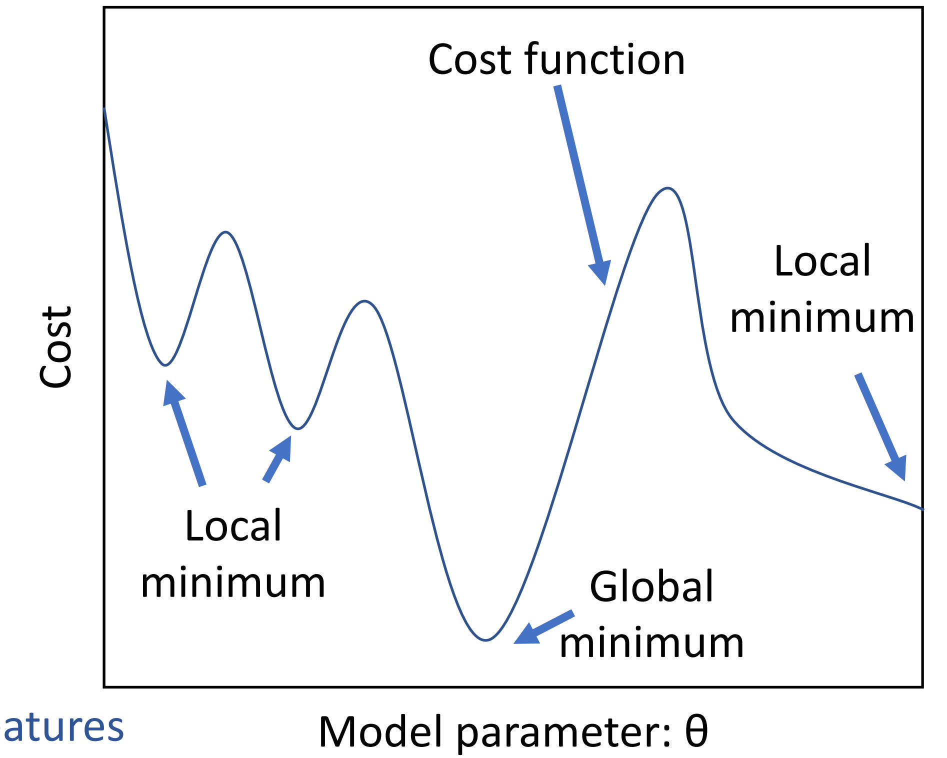


Image: Chartrand et al. RadioGraphics 2017

Cost Function = Error = Loss Function

- Measures errors or differences between predicted and target values
- Want to minimise it for training data

- Global minimum: smallest value overall
- Local minimum: smallest value in some region
- Example: Mean square error (MSE)



$$MSE(\boldsymbol{X}, h_{\boldsymbol{\theta}}) = \frac{1}{M}\sum_{i=1}^{M}\left(\boldsymbol{\theta}^T \boldsymbol{x}^{(i)} - y^{(i)}\right)^2$$

Input features for i^{th} *sample*

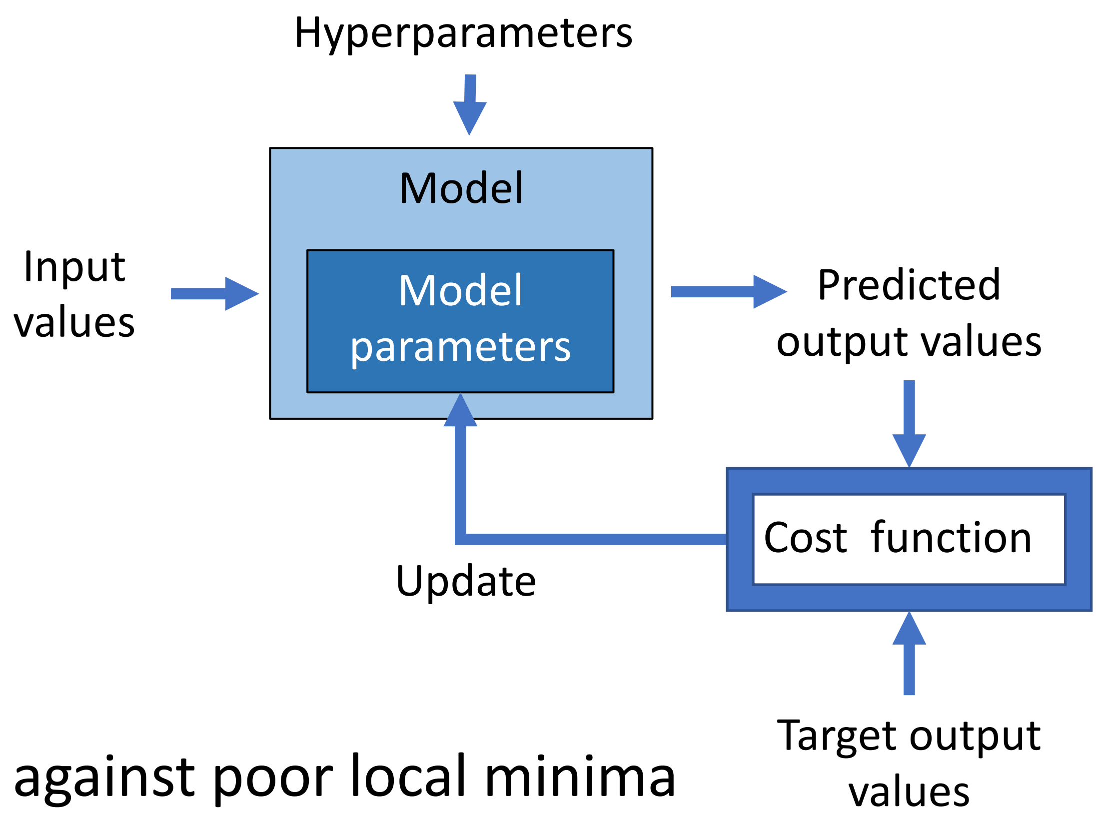
Targets for i^{th} *sample*

Parameters (same for all samples)

Training = Fitting = Optimisation

- Minimise cost function by adjusting model parameters

- Start with initial guess of model parameters
- Iteratively change model parameters & evaluate cost function



- Ideal algorithm: fast, but robust against poor local minima

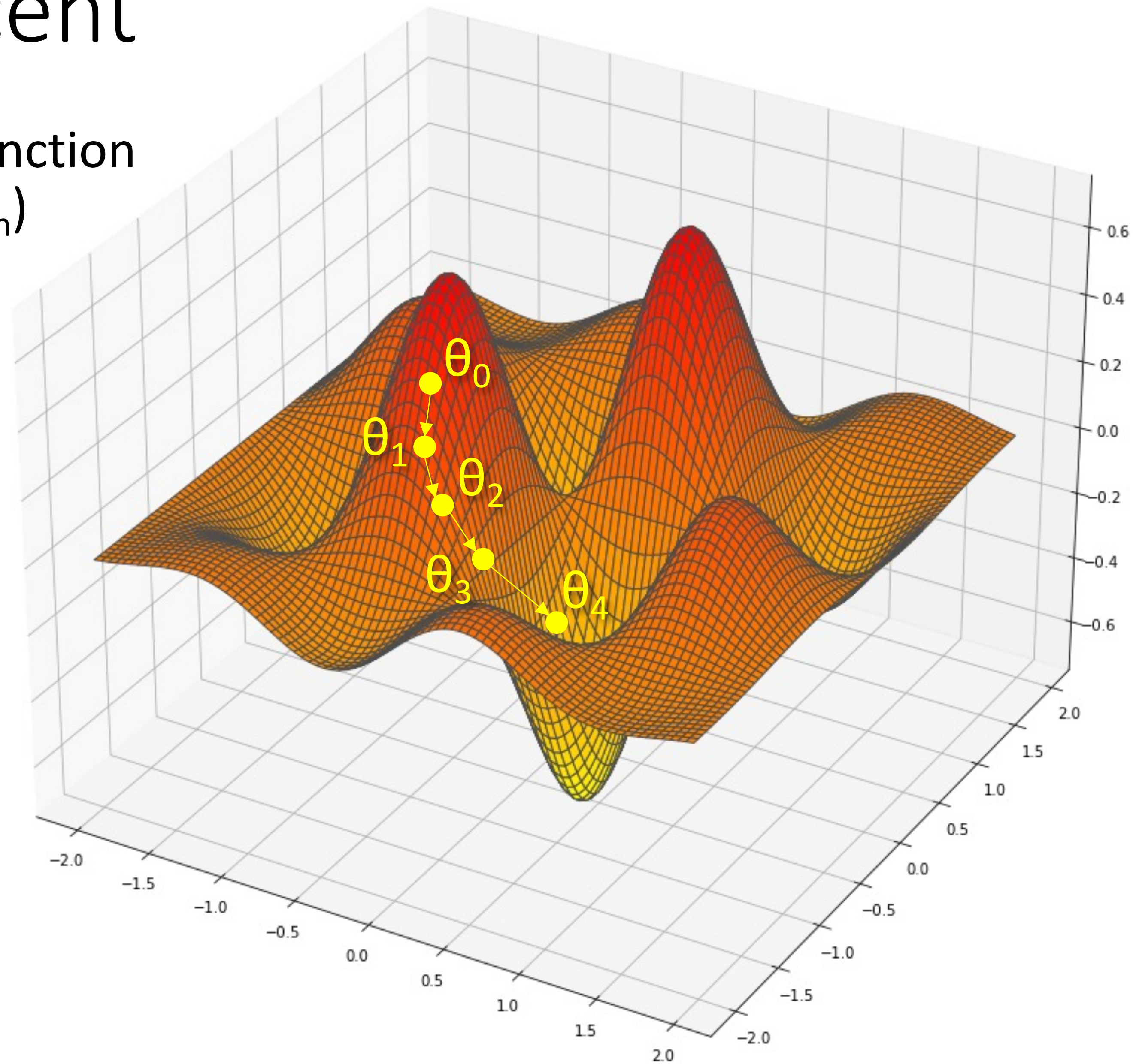
Gradient Descent

- Partial derivatives of cost function = Local gradient = $\nabla_\theta \text{ MSE}(\theta_n)$
- Iteratively step downhill (negative gradient)

$$\theta_{n+1} = \theta_n - \eta \nabla_\theta \text{ MSE}(\theta_n)$$

Learning rate "eta" η
= sets size of step

This is an important hyper-parameter!



Images: Geron, Hands On ML

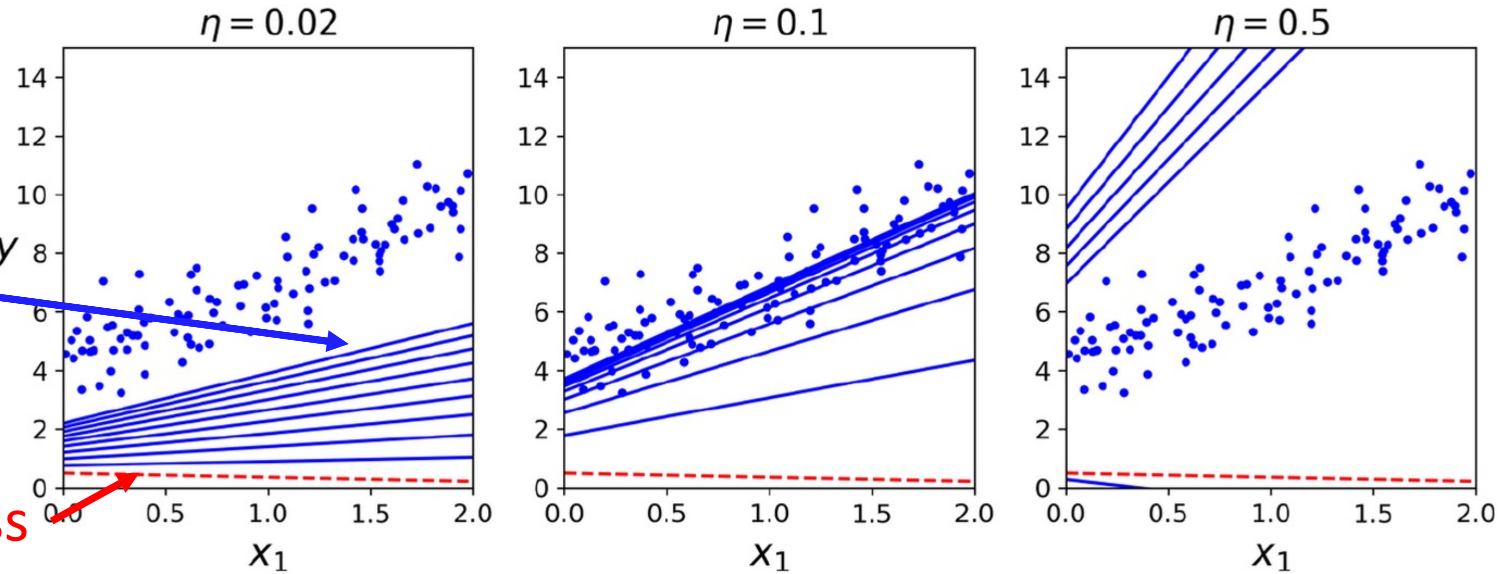
Learning rate η

Example: trying to fit line to points

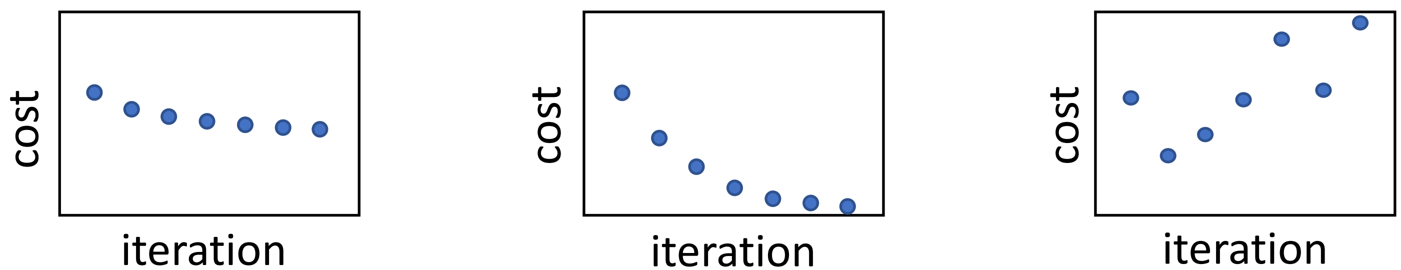
Too small | Good | Too large

n^{th} estimate

Initial guess



- Small η values take a long time to change, but go in the right direction
- Large η values overstep and can easily become unstable
- Can monitor the behaviour of the cost function values over iterations to spot these



Stopping Criteria and Final Result

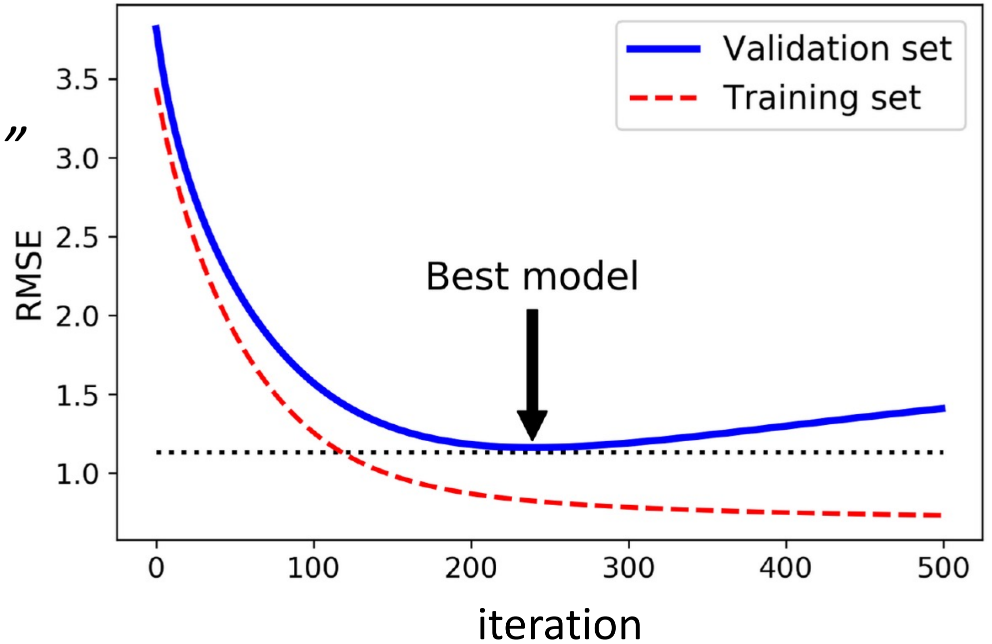
Stop if:

- No further improvement

  e.g. 5 iterations in a row show *"no change"*

  - `early_stopping`: turn on/off
  - `n_iter_no_change`: number of iterations
  - `tol`: if cost difference between steps is less than ε (tolerance) then treat it as *no change*

- Maximum number of iterations reached
  - `max_iter`: maximum number of iterations



- Final result: best model across the training process
  - this might *not* be the final model

Images: Geron, Hands On ML

Stochastic Gradient Descent

- Pick one random sample
- Calculate the cost function gradient only from that sample

Better algorithm:

- Shuffle instances of the training set
- Use one instance after the other
- Adjust the learning rate η
- Reshuffle and repeat

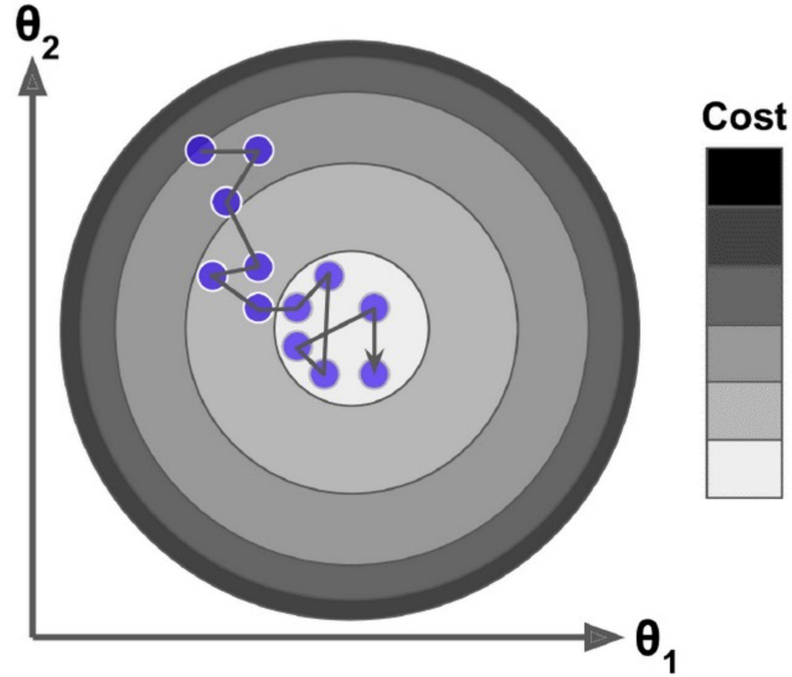
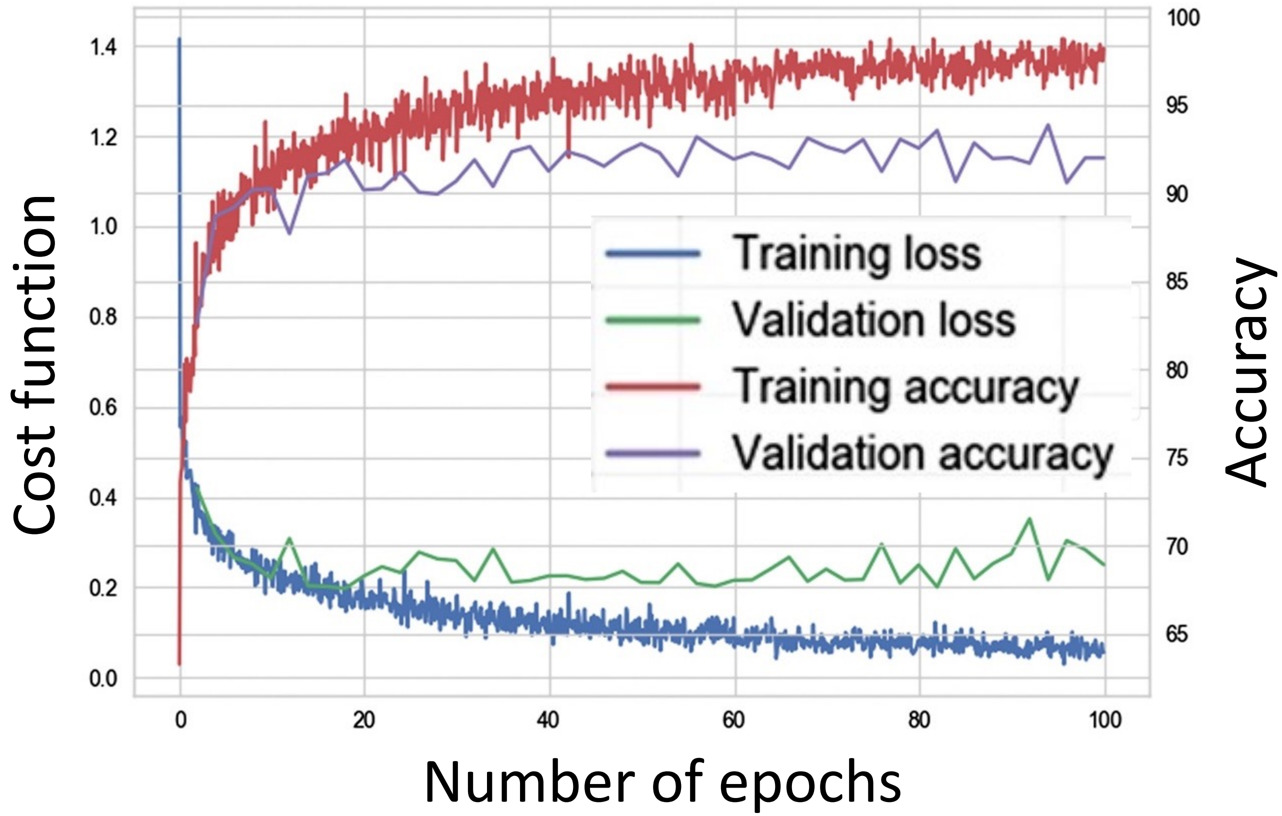


Image: Geron, Hands On ML

Pros: Fast, low memory, randomisation can help escape local minimum

Cons: Very noisy & no guarantee that minimum is reached

Learning Curve = Training Curve



- Epoch = one pass through whole dataset ≅ iteration
- Shows both training ***and*** validation performance
  - allows both underfitting and overfitting to be seen

Image: Chartrand et al. RadioGraphics 2017

Sources of Generalisation Error

- Variance
  - *Irreducible error*
    - Due to randomness in the data itself
  - *Overfitting* leads to over-sensitivity to small variations in the data
    - Too many model parameters

- Bias or systematic error
  - *Sub-optimal model choice or hyperparameter choice*
    - Especially underfitting
  - *Representativeness of data*
    - Lack of data coverage (model extrapolations are usually bad)
    - Bias in the data (e.g. due to limitations or bias in sampling)
    - Imbalances in the data (e.g. due to nature of problem)
      - disease vs healthy ; suspicious vs normal transactions

Regularisation

- Add a term to the cost function that tries to prevent overfitting
  - usually controlled by an adjustable weight α

Cost = data_term + α * regularisation_term

- Purpose
  - Prevent overfitting by penalising large parameter values or lack of smoothness in outputs
  - Add a-priori knowledge (desired properties) to an underdetermined problem
- A form of multi-objective optimisation

L2 (Ridge/Tikhonov) Regularisation

- Effect: Keep model parameters small

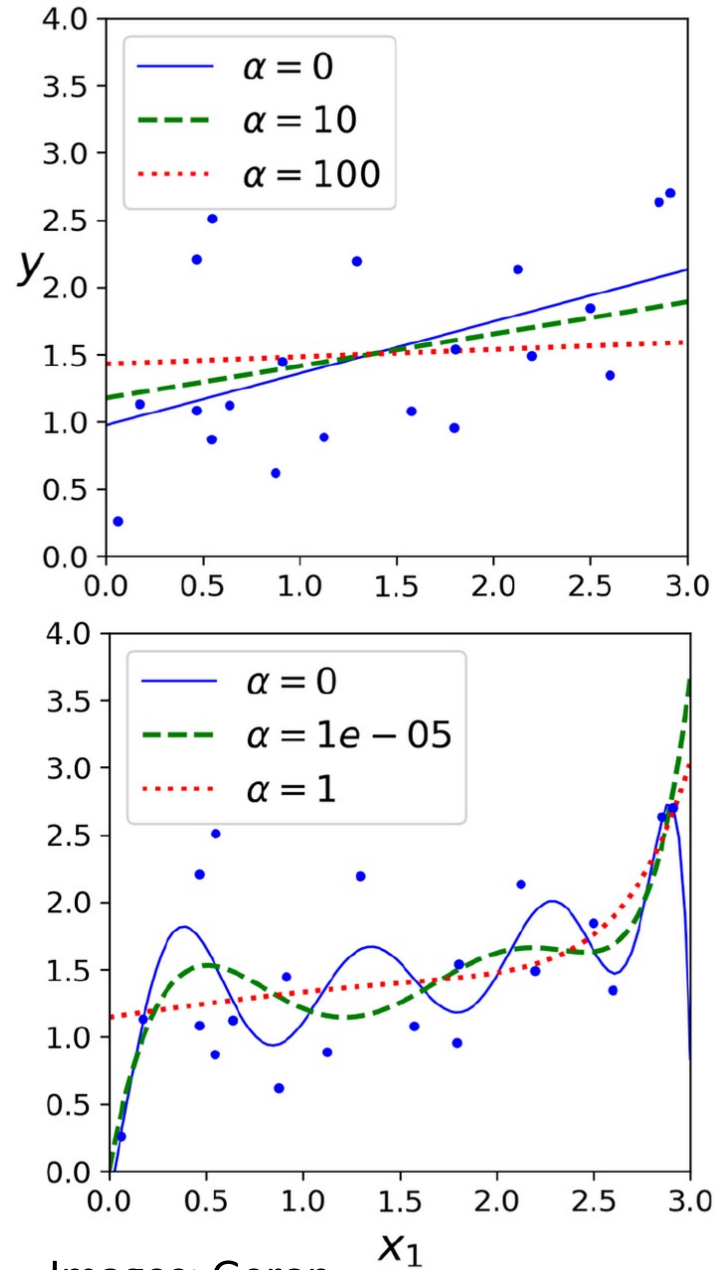
$$\overbrace{J(\boldsymbol{\theta})}^{\text{Cost}} = \overbrace{\text{MSE}(\boldsymbol{\theta})}^{\text{Data term}} + \alpha\overbrace{\tfrac{1}{2}\Sigma_{i=1}^{n}\,\theta_i^{\,2}}^{\text{Regularisation term}}$$

L2 Norm

Regularisation strength α

Model parameters, in this case except θ_0

- Scaling of data important for setting α
- Scikit learn: `penalty` parameter “`l2`”



Images: Geron, Hands On ML

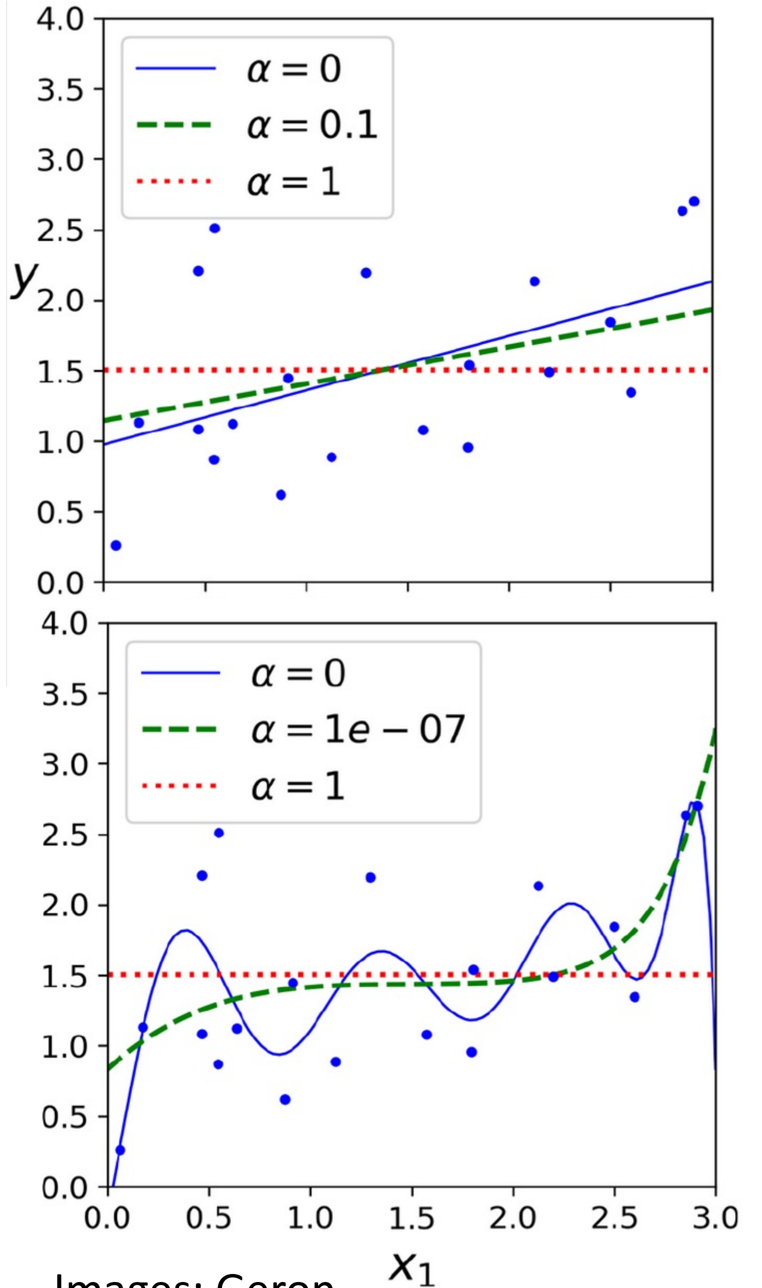
L1 (Lasso) Regularisation

- LASSO = Least Absolute Shrinkage and Selection Operator
- Effect: Keep model parameters small
- Also tends to eliminate least important features (i.e. sets some $\theta_i = 0$)

$$J(\boldsymbol{\theta}) = \mathrm{MSE}(\boldsymbol{\theta}) + \alpha \Sigma_{i=1}^{n} \left| \theta_i \right|$$

L1 Norm

- Scaling of data important for setting α
- Scikit learn: `penalty` parameter "`l1`"
- Not differentiable at 0, but most optimisers can cope with this



Images: Geron, Hands On ML

“Elastic Net”

- Just a mixture of L2 and L1 regularisation

$$\overbrace{J(\boldsymbol{\theta})}^{\text{Cost}} = \overbrace{\mathrm{MSE}(\boldsymbol{\theta})}^{\text{Data term}} + \overbrace{r\alpha\underbrace{\Sigma_{i=1}^{n}\left|\theta_i\right|}_{\text{L1}} + \frac{1-r}{2}\alpha\underbrace{\Sigma_{i=1}^{n}\theta_i^{2}}_{\text{L2}}}^{\text{Regularisation term}}$$

Mixture
ratio r

Summary

- Training: Minimise cost function by adjusting model parameters
- Regularisation: Penalise large parameters via an extra term in the cost function
- It can be helpful to understand how these work to get some intuition for good settings to use and how to diagnose problems
- Critically evaluate the implementation at hand and the defaults
    - Example: SGD classifier by default uses L2 (Ridge) regularisation