Different approaches to fitting and extrapolating the learning curve

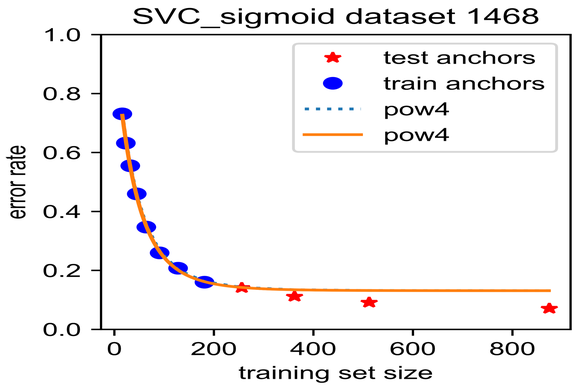
Donghwi Kim, Supervisor(s): Tom Viering, Marco Loog

TU Delft

Which approach gives the best performance for learning curve extrapolation?

- Curve fitting method
- Extrapolation

Leaning curve extrapolation



Divide empirical learning curve into train anchor and test anchor

Fit model function on empirical learning curve on training anchor

Evaluate extrapolation performance on test anchor

Model function example

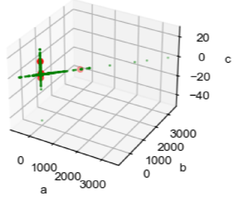
Model	Function
$log2$	$-a \log size + b$
$pow4$	$a - b * (size + d)^{-c}$

Curve fitting procedure

Parameter values initialisation The potential optimal parameter values used as starting point

Random initialisation given bound to each parameter

Distribution of optimal parameter values of pow3



From the optimal parameter values found from number of empirical learning curves, choose the k best initial parameter values by using K-means clustering algorithm (KMI)

Objective functions Function that curve fitting method aims to minimise

$$f(\vec{x}) = \sum_{i=0}^{n} (e_i - m_i(\vec{x}))^2 + \sum_{i=n+1}^{k} ((\max(1, m_i(\vec{x})) - 1) + \min(0, m_i(\vec{x}))^2$$

(1) MSE between empirical learning curve and extrapolated learning curve on training anchor

(2) Additional penalty if the range of error rate of extrapolated learning curve on test anchor is not between 0 or 1

i to *n*: index of the training sizes on train anchor

i+1 to *k*: index of the training sizes on test anchor

$\vec{x}$: vector containing parameter values *a, b, c*...

e_i : error rate from empirical learning curve on *i*th training size

$m_i(\vec{x})$: error rate estimated by model function using parameter values $\vec{x}$ on *i*th training size

Curve fitting methods Aims to find parameter values that minimise objective function

Newton

Aims to find the solution $\vec{x}$ such that $\nabla f(\vec{x}) = 0$ by using gradient and Hessian matrix of objective function. Note that $f(\vec{x})$ has local minimum at $\vec{x}$ where $\nabla f(\vec{x}) = 0$.

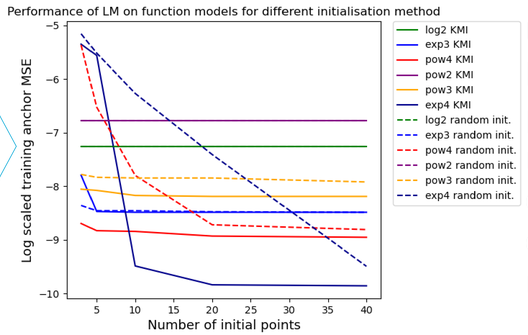
Levenberg–Marquardt (Gradient decent + Gauss–Newton)

Gradient decent (when far from optimal solution): finds optimal parameter values by updating parameter values in the steepest-descent direction.

Gauss-Newton (when close to optimal solution): minimises the sum of the squared errors by assuming the objective function locally quadratic and finds the minimum of quadratic.

Experiment results

Curve fitting performance



Using KMI showed better curve fitting performance than using random initial points especially for the complicated function models.

Extrapolation performance

LM,KMI,different obj. func.

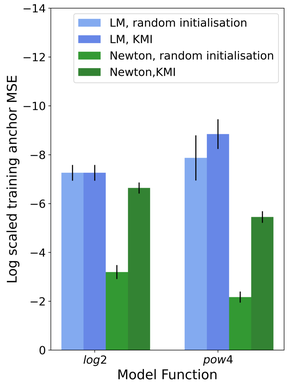
Objective func.		(1)	(2)
Bucket	Model		
0.05	log2	0.22	0.69
~0.1	pow4	0.68	0.82
0.1	log2	0.33	0.76
~0.2	pow4	0.8	0.92
0.2	log2	0.43	0.8
~0.4	pow4	0.89	0.91
0.4	log2	0.68	0.89
~0.8	pow4	0.94	0.94

Modifying objective function (2) showed some improvement on extrapolation performance compared to (1). It is also notable that pow4 always performs better than the log2.

(The result represents following. $N((e_k - m_k(\vec{x})) < 0.1)/N(e_k - m_k(\vec{x}))$

Bucket 0.1~0.2 means first 10~20% of empirical learning curve is used as train anchor and rest as test anchor)

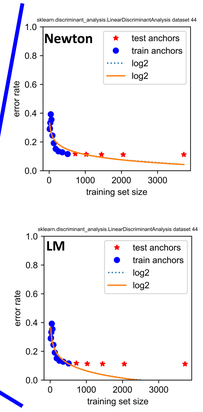
Curve fitting performance



Extrapolation performance KMI (1), different fitting

Fitting method		N	LM
Bucket	Model		
0.05	log2	0.32	0.22
~0.1	pow4	0.66	0.68
0.1	log2	0.38	0.33
~0.2	pow4	0.72	0.8
0.2	log2	0.43	0.43
~0.4	pow4	0.81	0.89
0.4	log2	0.54	0.68
~0.8	pow4	0.9	0.94

(N stands for Newton, LM stands for Levenberg–Marquardt)



Sometime, Better curve fitting ≠ Better extrapolation.

[1] Henri P. Gavin. The levenberg-marquardt method for nonlinear least squares curve-fitting problems c ©. 2013.

[2] Marco Loog Jan Van Rijn Felix Mohr, Tom Viering. Lcdb 1.0: An extensive learning curve database for classification tasks. ”under