Introduction to Machine Learning
Nonlinear Regression

Ramon Fuentes[1,2]

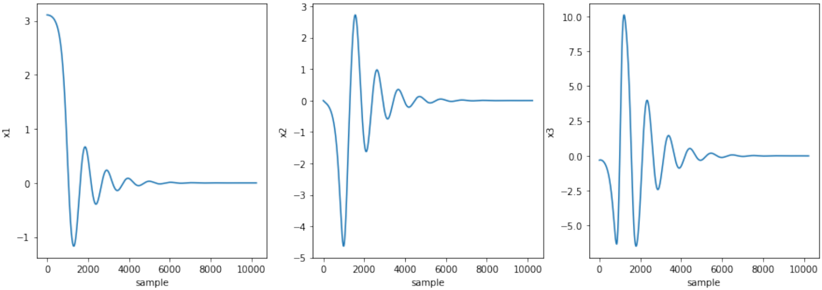
August 6, 2019

[1]Visiting Researcher, Dynamics Research Group
The University of Sheffield

[2]Research Scientist, Callsign Ltd

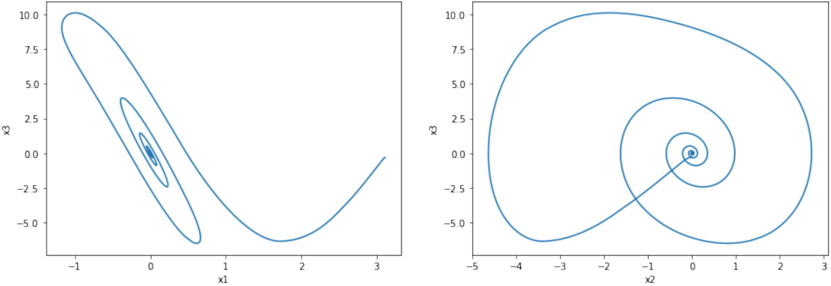
System identification with linear regression

Can we find the length of a pendulum given measured data from it?



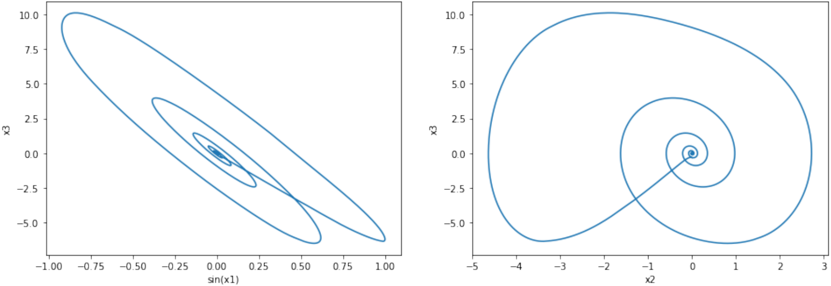
System identification with linear regression

The relationship between x_1 and x_3 is nonlinear



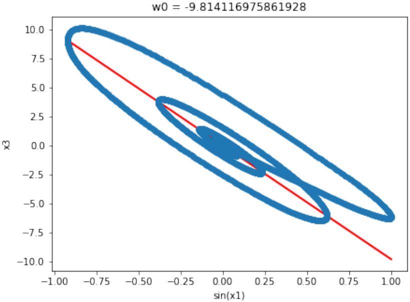
System identification with linear regression

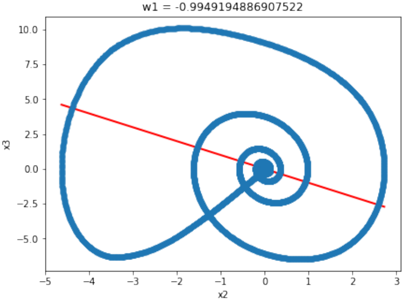
but what if we simply transform it so that a linear relationship holds?



System identification with linear regression

Applying linear regression to it, we can recover the length and damping coefficient!





Linear parameter models

Linear regression can be used to solve complex nonlinear problems, by transforming the data so that it is linear in some domain

- These are called linear parameter / linear-in-the-parameter models

Linear parameter models

Linear regression can be used to solve complex nonlinear problems, by transforming the data so that it is linear in some domain

- These are called linear parameter / linear-in-the-parameter models
- In principle, we could use complex transforms of our input data to suit the problem at hand. For example:

$$\mathbf{X} = [1, \mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_0\mathbf{x}_1, \mathbf{x}_0^2, \mathbf{x}_1^2, ..., \sin(\mathbf{x}), \cos(\mathbf{x}), \text{sign}(\mathbf{x}), ...]$$

Linear parameter models

Linear regression can be used to solve complex nonlinear problems, by transforming the data so that it is linear in some domain

- These are called linear parameter / linear-in-the-parameter models
- In principle, we could use complex transforms of our input data to suit the problem at hand. For example:

$$\mathbf{X} = [1, \mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_0\mathbf{x}_1, \mathbf{x}_0^2, \mathbf{x}_1^2, ..., \sin(\mathbf{x}), \cos(\mathbf{x}), \text{sign}(\mathbf{x}), ...]$$

- we're only limited by our imaginations

Linear parameter models

Linear regression can be used to solve complex nonlinear problems, by transforming the data so that it is linear in some domain

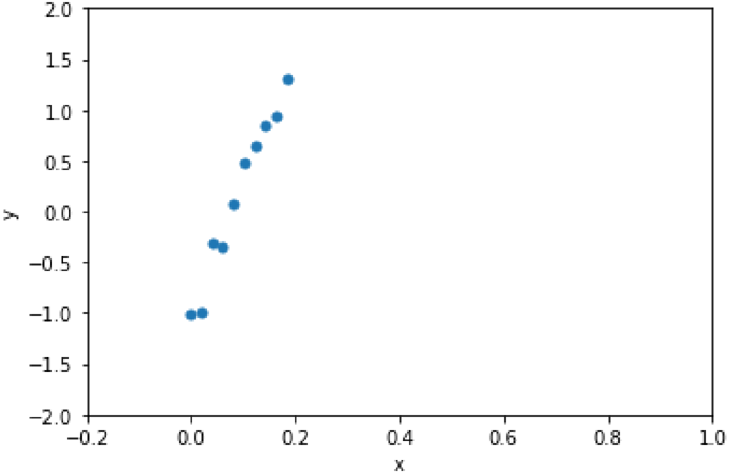
- These are called linear parameter / linear-in-the-parameter models
- In principle, we could use complex transforms of our input data to suit the problem at hand. For example:

$$\mathbf{X} = [1, \mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_0\mathbf{x}_1, \mathbf{x}_0^2, \mathbf{x}_1^2, ..., \sin(\mathbf{x}), \cos(\mathbf{x}), \text{sign}(\mathbf{x}), ...]$$

- we're only limited by our imaginations
- but life is not that simple...

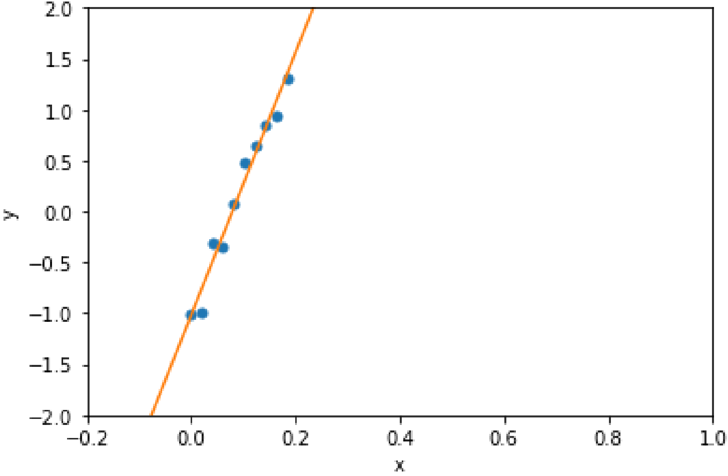
Polynomial expansions

Lets look at another example...



Polynomial expansions

Looks linear, so lets fit a linear model $\mathbf{y} = [1, \mathbf{x}]\mathbf{w}$

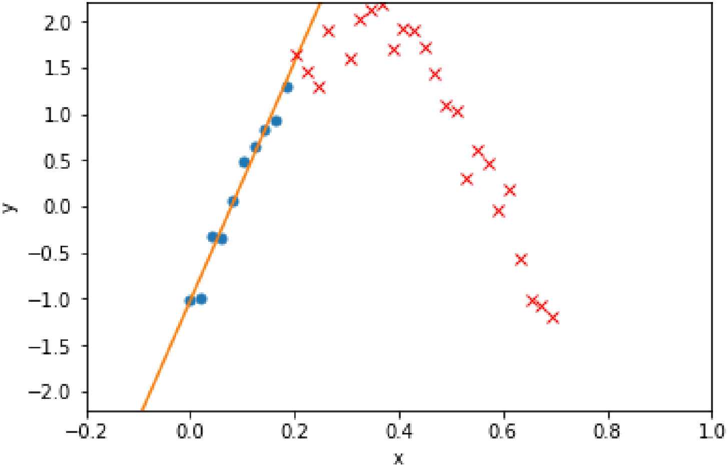


Polynomial expansions

great! but how good are these predictions elsewhere? lets get some more data...

Polynomial expansions

great! but how good are these predictions elsewhere? lets get some more data...

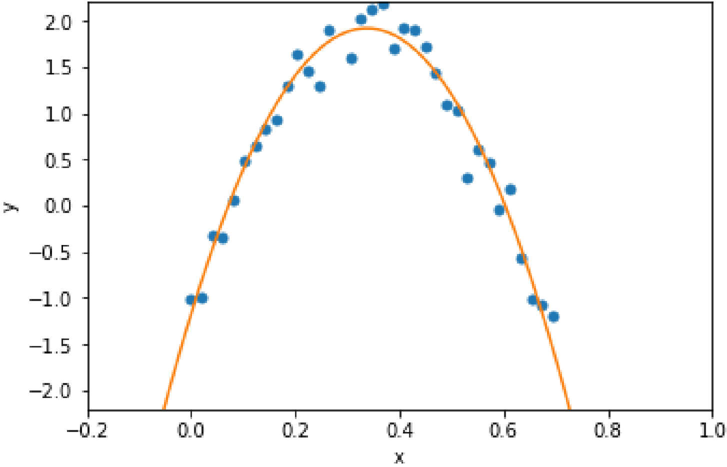


Polynomial expansions

oh no! ...but hold on, we can fit a quadratic polynomial to this,

Polynomial expansions

oh no! ...but hold on, we can fit a quadratic polynomial to this,

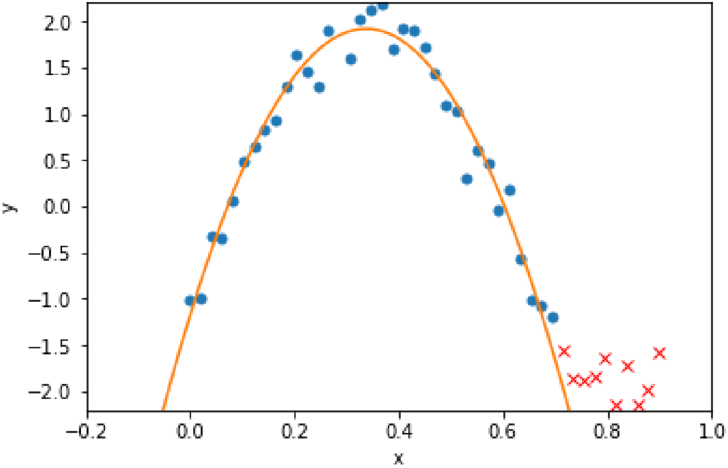


Polynomial expansions

And how good is this at predicting outside the training region?

Polynomial expansions

And how good is this at predicting outside the training region?

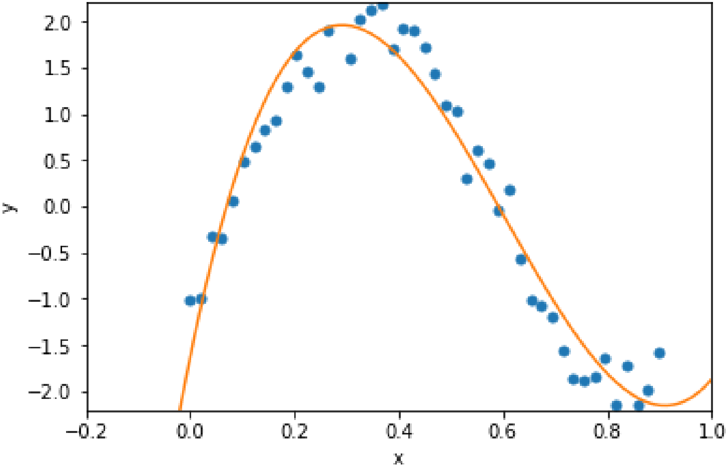


Polynomial expansions

Ok... we can fit a 3^{rd} order polynomial...

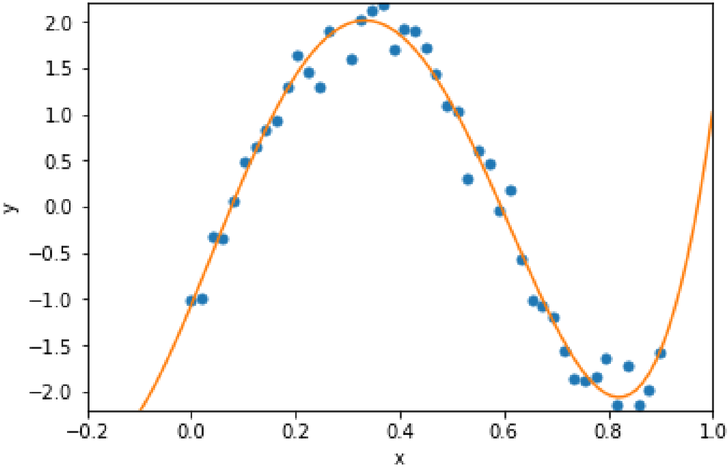
Polynomial expansions

Ok... we can fit a 3^{rd} order polynomial...



Polynomial expansions

and a fourth order...



Polynomial expansions

A few question arise:

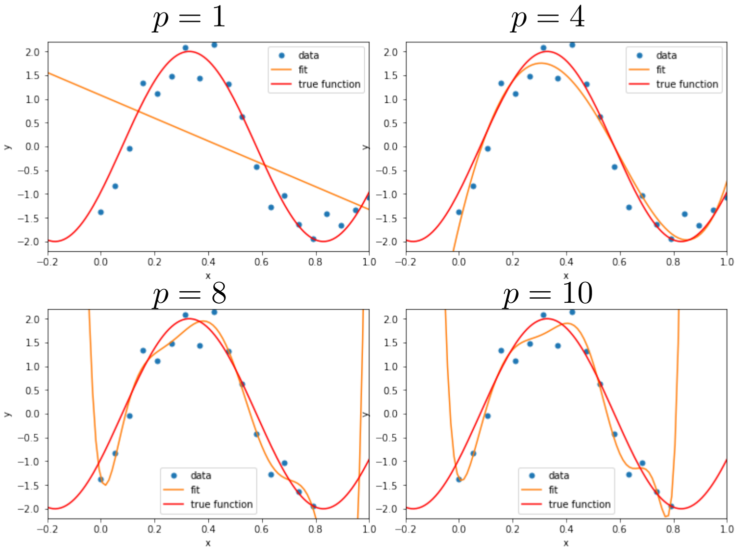
- Should we keep increasing the complexity of the models to minimise the training error?
- At what point do we stop?
- How do we assess model performance outside of the region covered by training data ?

Model Complexity

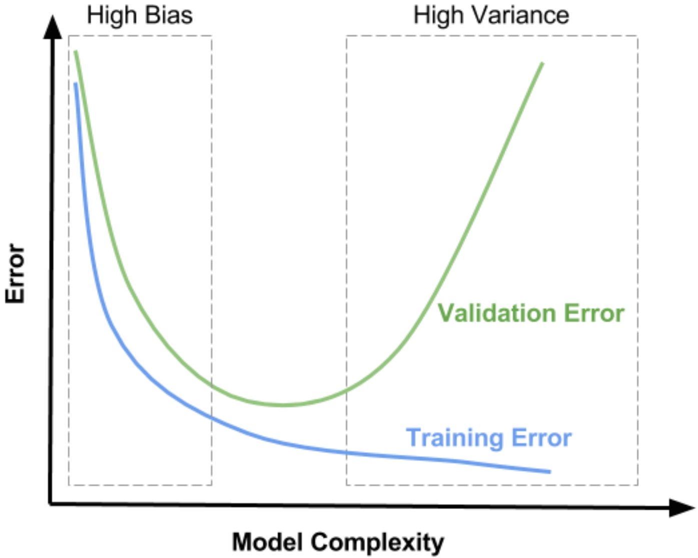
By now, we will have noticed that

- Simple models don't perform that well on complex data
- Complex models perform well on the data that they've been trained on, but fail to accurately predict outside that range. They **over-fit**

Bias, Variance and overfitting



Bias and variance



Bias and variance

- Simple models underfit / have high bias / high training error
- Complex models overfit / have high variance / high generalisation error
- A balance is needed!

Bias and variance

We have two main tools to balance model complexity and quality of fit:

- Regularisation
- Cross-validation

Regularisation

One way to achieve a balance of complexity and quality of fit is to penalise more complex models through additional terms in the loss function.

In linear regression, a popular penalty is:

$$J(\mathbf{w}) = ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \lambda||\mathbf{w}||_p \tag{1}$$

Note that this penalty can also be interpreted as a constraint on the loss function

Regularisation

$$J(\mathbf{w}) = ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \lambda||\mathbf{w}||_p \qquad (2)$$

The norm, p, in the loss function plays an important role on the type of regularisation.

- $p = 0$ leads to a combinatorial selection of one candidate model amongst all, and is generally hard to optimise.

Regularisation

$$J(\mathbf{w}) = ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \lambda||\mathbf{w}||_p \quad (2)$$

The norm, p, in the loss function plays an important role on the type of regularisation.

- $p = 0$ leads to a combinatorial selection of one candidate model amongst all, and is generally hard to optimise.
- $p = 1$ leads to a sparse solution over the weight vector, $\mathbf{w}$. This is known as Lasso regression. There is no closed-form solution, but practical optimisation algorithms exist for this

Regularisation

$$J(\mathbf{w}) = ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \lambda||\mathbf{w}||_p \quad (2)$$

The norm, p, in the loss function plays an important role on the type of regularisation.

- $p = 0$ leads to a combinatorial selection of one candidate model amongst all, and is generally hard to optimise.
- $p = 1$ leads to a sparse solution over the weight vector, $\mathbf{w}$. This is known as Lasso regression. There is no closed-form solution, but practical optimisation algorithms exist for this
- $p = 2$ leads to circular constraint over the loss function, and a closed form solution exists for this penalty!

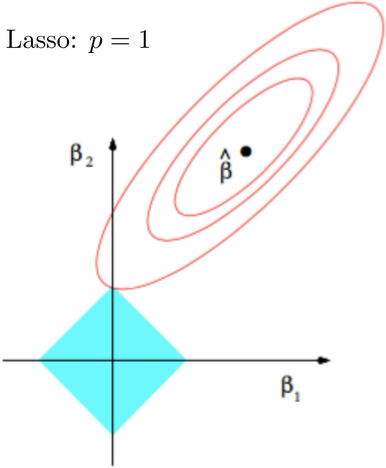
Regularisation

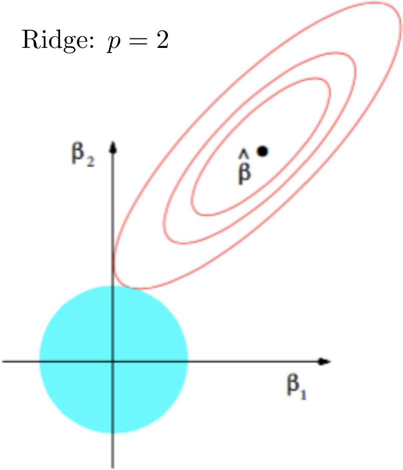
$$J(\mathbf{w}) = ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \lambda||\mathbf{w}||_p \qquad (2)$$

The norm, p, in the loss function plays an important role on the type of regularisation.

- $p = 0$ leads to a combinatorial selection of one candidate model amongst all, and is generally hard to optimise.
- $p = 1$ leads to a sparse solution over the weight vector, $\mathbf{w}$. This is known as Lasso regression. There is no closed-form solution, but practical optimisation algorithms exist for this
- $p = 2$ leads to circular constraint over the loss function, and a closed form solution exists for this penalty!
- We'll focus on $p = 2$ here, otherwise known as Tikhonov reugarisation

Regularisation constraints of Lasso and Ridge regression





Regularisation and ill-posedness

- Regularisation not only helps balance model complexity,

Regularisation and ill-posedness

- Regularisation not only helps balance model complexity,
- it also helps to better condition ill-posed inverse problems

Regularisation and ill-posedness

- Regularisation not only helps balance model complexity,
- it also helps to better condition ill-posed inverse problems
- OLS solution involves the inversion: $(\mathbf{X}^T\mathbf{X})^{-1}$

Regularisation and ill-posedness

- Regularisation not only helps balance model complexity,
- it also helps to better condition ill-posed inverse problems
- OLS solution involves the inversion: $(\mathbf{X}^T\mathbf{X})^{-1}$
- the solution to it, factorisation might be numerically unstable if:

Regularisation and ill-posedness

- Regularisation not only helps balance model complexity,
- it also helps to better condition ill-posed inverse problems
- OLS solution involves the inversion: $(\mathbf{X}^T\mathbf{X})^{-1}$
- the solution to it, factorisation might be numerically unstable if:
    - there are significantly more bases than observations

Regularisation and ill-posedness

- Regularisation not only helps balance model complexity,
- it also helps to better condition ill-posed inverse problems
- OLS solution involves the inversion: $(\mathbf{X}^T\mathbf{X})^{-1}$
- the solution to it, factorisation might be numerically unstable if:
    - there are significantly more bases than observations
    - the bases/columns in $\mathbf{X}$ are not linearly independent (solution is not unique)

Tikhonov regularisation

- We have derived the solution to the basic least squares linear regression

Tikhonov regularisation

- We have derived the solution to the basic least squares linear regression
- Tikhonov regularisation follows closely from it

Tikhonov regularisation

- We have derived the solution to the basic least squares linear regression
- Tikhonov regularisation follows closely from it
- Our loss function has an additional term, $||\mathbf{w}||_2^2$

Tikhonov regularisation

- We have derived the solution to the basic least squares linear regression
- Tikhonov regularisation follows closely from it
- Our loss function has an additional term, $||\mathbf{w}||_2^2$
- and we have that $\nabla||\mathbf{w}||_2^2 = 2\mathbf{w}$

Deriving Tikhonov regularisation

We need to minimise:

$$J(\mathbf{w}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda||\mathbf{w}||_2^2$$

Deriving Tikhonov regularisation

We need to minimise:

$$J(\mathbf{w}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda||\mathbf{w}||_2^2$$

We have that,

$$\nabla_{\mathbf{w}}||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 = 2\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{X}^T\mathbf{y}$$

Deriving Tikhonov regularisation

We need to minimise:

$$J(\mathbf{w}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda||\mathbf{w}||_2^2$$

We have that,

$$\nabla_{\mathbf{w}}||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 = 2\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{X}^T\mathbf{y}$$

and also,

$$\nabla_{\mathbf{w}}||\mathbf{w}||_2^2 = 2\lambda\mathbf{w}$$

Deriving Tikhonov regularisation

We need to minimise:

$$J(\mathbf{w}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda||\mathbf{w}||_2^2$$

We have that,

$$\nabla_{\mathbf{w}}||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 = 2\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{X}^T\mathbf{y}$$

and also,

$$\nabla_{\mathbf{w}}||\mathbf{w}||_2^2 = 2\lambda\mathbf{w}$$

so,

$$\mathbf{X}^T\mathbf{y} = \mathbf{X}^T\mathbf{X}\mathbf{w} + \lambda\mathbf{w}$$

Deriving Tikhonov regularisation

We need to minimise:

$$J(\mathbf{w}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda||\mathbf{w}||_2^2$$

We have that,

$$\nabla_{\mathbf{w}}||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 = 2\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{X}^T\mathbf{y}$$

and also,

$$\nabla_{\mathbf{w}}||\mathbf{w}||_2^2 = 2\lambda\mathbf{w}$$

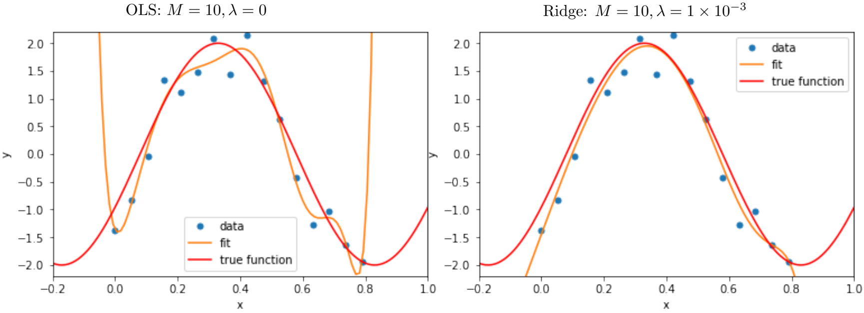
so,

$$\mathbf{X}^T\mathbf{y} = \mathbf{X}^T\mathbf{X}\mathbf{w} + \lambda\mathbf{w}$$

rearranging for $\mathbf{w}$

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

Regularisation - Example



Regularisation

Life is good, but have we replaced one problem with another?

- The regularisation coefficient, λ now balances model complexity
- We need an effective method for selecting it, based on generalisation performance

Generalisation error

- In order to assess generalsation error, we must set aside a sample of our training data for evaluation

Generalisation error

- In order to assess generalsation error, we must set aside a sample of our training data for evaluation
- However, training data can be scarce! Holding out data means it does not inform training... so maybe not a good idea?

Generalisation error

- In order to assess generalsation error, we must set aside a sample of our training data for evaluation
- However, training data can be scarce! Holding out data means it does not inform training... so maybe not a good idea?

Conclusions

What have we learned today?

- How to do nonlinear regression, using linear regression
- Generalisation
- The bias-variance trade-off - balancing model complexity
- Regularisation

So... what next?

Tomorrow, we'll learn about some even more flexible models for regression, and how to tune hyper-parameters through cross-validation ;)