EngSci 721

Inverse Problems and Learning From Data

Oliver Maclaren (oliver.maclaren@auckland.ac.nz)

1. *Basic concepts* [5 lectures + 1 Tutorial]

   Forward vs inverse problems. Well-posed vs ill-posed problems. Algebra and calculus of inverse problems (left and right inverses, generalised and pseudo inverses, resolution operators, matrix calculus). Representing higher dimensional problems (image data etc).

2. *Instability and regularisation in linear and nonlinear problems* [6 lectures + 1 Tutorial]

   Instability and related issues for generalised inverses. Introduction to regularisation and trade-offs. Tikhonov regularisation. Higher-order Tikhonov regularisation. Sparsity and regularisation using different norms. Truncated singular value decomposition. Iterative regularisation, including stochastic/mini-batch gradient descent.

3. *Further topics* [3 lectures + 1 Tutorial]

   Regularisation parameter choice, including statistical and machine learning views of regularisation. Confidence sets for linear and nonlinear models. Physics-informed machine learning and neural networks.

Module overview

Inverse Problems and Learning From Data (*Oliver Maclaren*)
[~14 lectures/3 tutorials]

Lecture 6: Introduction to Tikhonov Regularisation

Topics:

- Regularisation in general
- Tikhonov regularisation
- Examples

Eng Sci 721 : Lecture 6

Intro to (Tikhonov) regularisation.

→ We have seen that generalised* inverses give minimum norm least squares solutions

→ We have also seen that this solves existence & uniquess issues, but does not solve stability issues. eg →

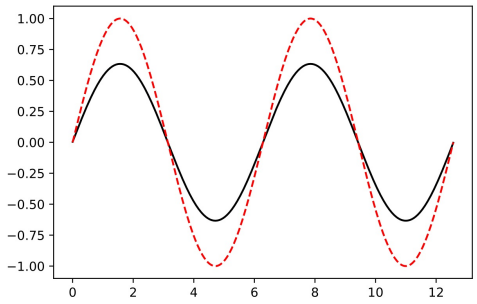
What's going on?

How can we do better?..

* really, psendo inverses (all four conditions)

Recall : instability of $A^{\dagger}$ ($= A^{-1}$, if exists)

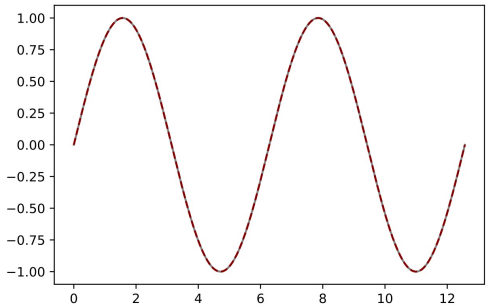
Smoothing map :



red : original
black : local av.

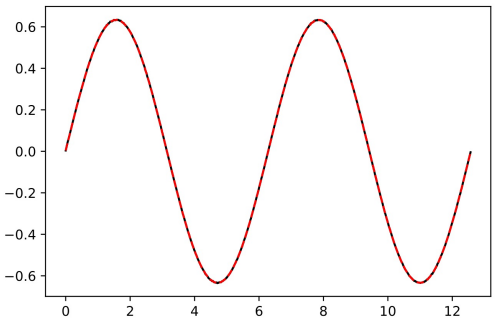
Can implement as linear forward map, ie matrix A (exercise?)

Invert noise free gives :



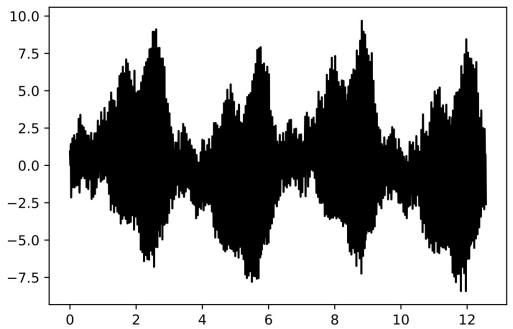
Recall: instability of A^+ ($= A^{-1}$, if exists)

now, add small noise:



practically indetectable!

Invert noisy using A^{-1}



Terrible!

How to 'stabilise': Regularisation!

In the linear setting we have A^+ (gen. inverse) but it is possibly discontinuous / unbounded / ill-conditioned etc

↳ same issue for nonlinear (can define gen. inv too)

→ we would usually rather have an 'approximate generalised inverse' that is continuous / bounded / well-conditioned

→ A 'regularisation method' is any approach to constructing a continuous / well-behaved but approximate generalised inverse (a 'regularised inverse')

Approaches

- optimisation/variational { Tikhonov ← today
- algebraic { approximate factorisations } later
  - eg truncated SVD.

Recall: Can characterise A^+ via
- algebra
  - eg $A^+AA^+ = A^+$ etc
- optimisation
  - eg $\min \|\|^2$
  - s.t. $\min \|\|^2$
  - etc.

Optimisation formulation revisited

Recall:

We can think of the generalised inverse as the result of a two-step optimisation problem:

1. Find the set of least squares solutions
2. Of these solutions, choose the minimum norm solution.

It turns out that it is better to simultaneously control these two objectives & trade-off
- fit to data
- model 'complexity' (size)

Example : fitting a polynomial to data

$$y = a_0 + a_1 x + a_2 x^2 + \dots a_n x^n$$

observations:

$$(x_0, y_0), (x_1, y_1), \dots (x_m, y_m)$$

Leads to

$$y_0 = a_0 + a_1 x_0 + a_2 x_0^2 + \dots a_n x_0^n$$

$$y_1 = a_0 + a_1 x_1 + a_2 x_1^2 + \dots a_n x_1^n$$

$$\vdots$$

$$y_m = a_0 + a_0 x_m + a_2 x_m^2 + \dots a_n x_m^n$$

ie

$$\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_m \end{pmatrix} = \begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \vdots & & & & \\ 1 & - & - & - & x_m^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix}$$

ie $y_{obs} = A_{obs}\,\theta$ linear system

→ linear in parameters !

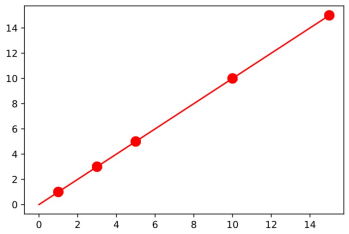
Underdetermined case & instability

Eg - high degree polynomial } possibly non unique.
- plus observation noise } possibly inconsistent

$$y = a_0 + a_1 x + a_2 x + \dots a_n x^n$$ class

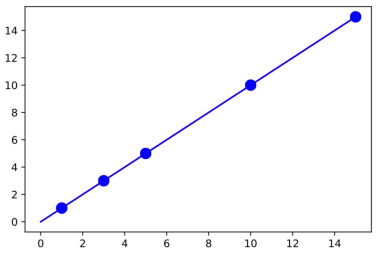
But true model: $a_i = 0$ if $i \neq 1$
$a_1 = 1$

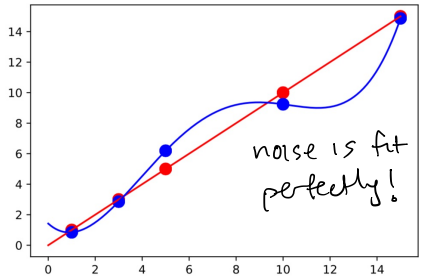
'True' data: (5 observations)



noise free recovery

add noise & recover





but see later on 'double descent'!

unstable! Not reduced enough!
→ fitting noise
(not reproducible!)

What we seem to want is something that generally fits the data less well

→ Huh?

We are willing to trade-off some fit to given data for a 'simpler' or more stable model

↳ 'similar' data → similar model

Statistical/predictive view:

- want to fit out-of-sample as well as in-sample data

↳ training vs test sets & overfitting

↳ noise is not exactly reproducible, hence don't try to fit it exactly!

↳ 'bias-variance trade-off'

ML note: recent results on good interpolating models! } 'double descent'
↳ we'll look at?

Trade-offs & regularisation

- Instead of two-step:
  - Stage 1: minimise $\|y - Ax\|$ or $\|y - Ax\|^2$ x
  - Then
  - Stage 2: minimise $\|x\|$ or $\|x\|^2$ among all solutions to stage 1.

- Try simultaneous minimisation:

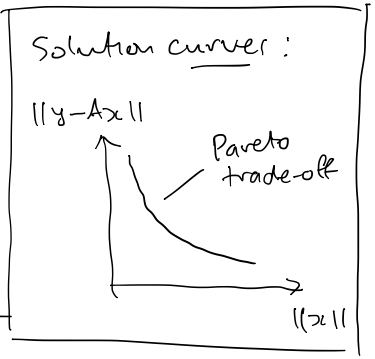
vector/multiobjective problem

$$\min_x \left(\|y - Ax\|, \|x\| \right)$$

simultaneously

→

Solution curves:



→ allows us to filter noise in underdetermined case, while still 'shrinking' or reducing models to get 'simple' solutions

Stepwise revisited

We have seen

1. First fit data
2. Then reduce model

Leads to the (possibly) unstable generalised inverse.

Suppose we instead tried

1. First reduce model
2. Then fit data.

→ suppose $\min\{\|x\|\} = 0$ & $x = 0$ is the soln.

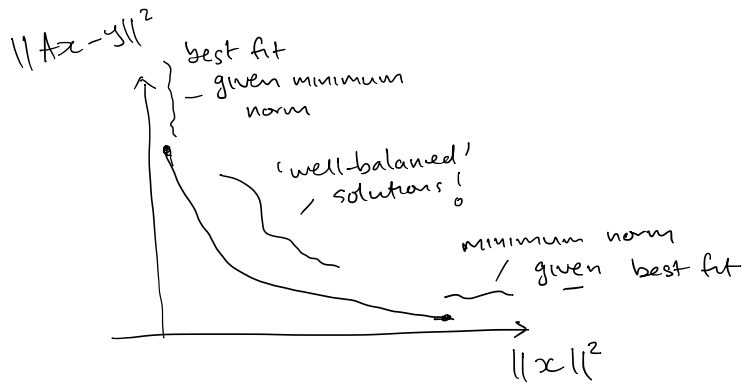
Then for $Ax = y$

→ data error is

$$\|y - Ax\| = \|y\|$$

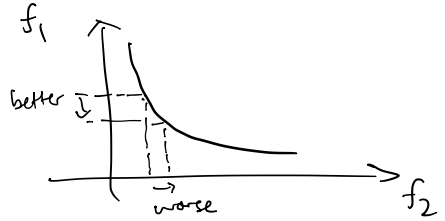
→ not very good either!

Trade-off curve : limits/end points



Pareto solutions : sets of solutions

- equally good, unless we prefer one objective to other
- can't improve one objective without making other worse:



Three (or more) equivalent versions

The bi-objective problem can be written as a standard single objective problem in at least 3 equivalent ways

→ The idea is to introduce an extra parameter that, one way or the other, represents the relative importance, or trade-off, for the two objectives

L Think: 'exchange rate' etc.

L By varying the exchange rate we 'trace out' all the solutions on the Pareto curve

L in particular we control the balance, rather than prioritise one over other as in 2-step.

Version 1: simple model, acceptable fit.

1a. $\min \|x\|$ s.t. $\|Ax - y\| \leq \delta$

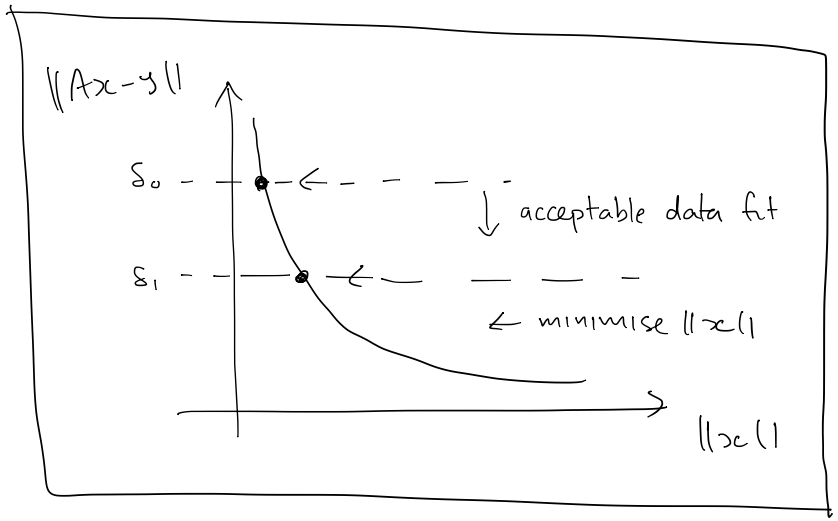
smallest model that fits to within δ (not zero!)

or

1b. $\min \|x\|^2$ s.t. $\|Ax - y\|^2 \leq \delta$ etc.

Parameterised by δ,

ie 'acceptable data fit' level:



Version 2: acceptable model, best fit

2a. $\min \|Ax - y\|$ s.t. $\|x\| \leq \epsilon$

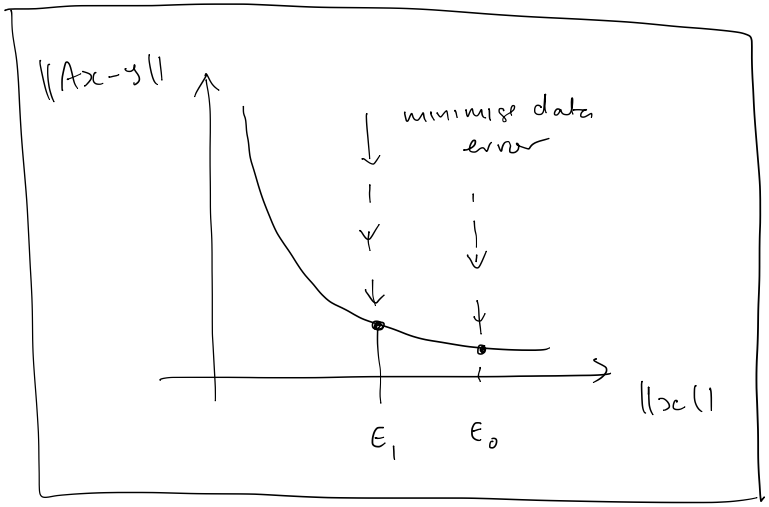
best fit within acceptable model limits (not too 'complex')

or

2b. $\min \|Ax - y\|^2$ s.t. $\|x\|^2 \leq \epsilon$ etc.

Parameterised by ϵ,

ie 'allowable model complexity':



Version 3 : weighted sum ('scalarised' version)

3a. $\boxed{\min \|Ax-y\| + \lambda \|x\|}$, $\lambda > 0$

3b. $\boxed{\min \|Ax-y\|^2 + \lambda \|x\|^2}$ (or $\min \lambda^T f$ for vector of weights & f vector obj.)

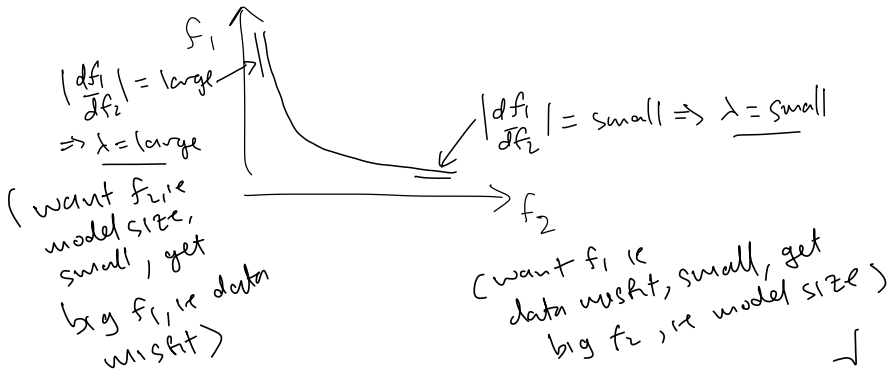
3b': { "Tikhonov" regularisation, Damped least squares, Ridge regression } etc!

→ Solve for various λ: 'trade-off weight', 'relative importance'

$L = f_1 + \lambda f_2$

$dL = df_1 + \lambda df_2 = 0$

$\Rightarrow \lambda = -\frac{df_1}{df_2} = \left|\frac{df_1}{df_2}\right|$



Analytical solution (linear only)

The form $\boxed{\|Ax-y\|^2 + \lambda \|x\|^2}$ is particularly useful for getting an analytical soln

Note : $\|r\|^2 = \|r_1\|^2 + \|r_2\|^2$, for $r = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}$ (sub vectors)

⇒ define the augmented/stacked vector:

$$\tilde{r} = \begin{bmatrix} Ax-y \\ \sqrt{\lambda} x \end{bmatrix} \Rightarrow \|\tilde{r}\|^2 = \|Ax-y\|^2 + \lambda \|x\|^2$$

This can be written as

$$\tilde{r} = \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix} x - \begin{bmatrix} y \\ 0 \end{bmatrix}$$

$$= \tilde{A}x - \tilde{y}$$, ie augmented/stacked residual equation

→ can solve via ordinary least squares!

We want to solve the augmented/stacked least squares problem:

$$\min \|\tilde{r}\|^2 = \min \|\tilde{A}x - \tilde{y}\|^2$$

$$\left(\text{i.e. } \min \left\| \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix} x - \begin{bmatrix} y \\ 0 \end{bmatrix} \right\|^2 \right)$$

→ can we?

Note: Ix & $x \in \mathbb{R}^n$ means

I is $n \times n$ identity

A is $m \times n$, where $m < n$ (unders.

so $\tilde{A} = \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix} = \begin{bmatrix} m \times n \\ \\ n \times n \\ \\ \\ \end{bmatrix}$ $m+n > n$

n

⇒ now have tall, over-determined system.

Least squares solution exists if

- columns are LI
- $\tilde{A}^T\tilde{A}$ is invertible

} equiv.

etc

Key:

> It turns out that the columns of $\tilde{A}$ are LI & $\tilde{A}^T\tilde{A}$ is invertible, regardless of A, as long as $\lambda > 0$!

→ (we'll prove in tutorial?)

note: $\begin{bmatrix} \\ \end{bmatrix}$, $\begin{bmatrix} \vdots \\ 0 \end{bmatrix}$ } LI

---

so the normal equations

$\tilde{A}^T\tilde{A}x = \tilde{A}^T\tilde{y}$ are always solvable

for

$$x = \tilde{A}^+\tilde{y}$$
$$= (\tilde{A}^T\tilde{A})^{-1}\tilde{A}^T\tilde{y}$$

→

Regularised normal equations

Note:

$$\tilde{A}^T\tilde{A} = \begin{bmatrix} A^T & \sqrt{\lambda}I \end{bmatrix}\begin{bmatrix} A \\ \sqrt{\lambda}I \end{bmatrix} = (A^TA + \lambda I)$$

&

$$\tilde{A}^T\tilde{y} = \begin{bmatrix} A^T & \sqrt{\lambda}I \end{bmatrix}\begin{bmatrix} y \\ 0 \end{bmatrix}$$

$$= A^Ty$$

So the normal equations become

$$\boxed{(A^TA + \lambda I)x = A^Ty}$$

↑ only extra term.

Explains why works:

A^TA is positive semi-definite } almost invertible

λI is positive definite, $\lambda > 0$

add together gives positive definite → invertible!

Limiting cases

consider the 'Tikhonov' inverse in

$$x = (A^TA + \lambda I)^{-1}A^Ty$$

ie $\boxed{x = A_\lambda^* y}$ where $\boxed{A_\lambda^* = (A^TA + \lambda I)^{-1}A^T}$

Cases.

1. As $\lambda \to 0$, keeping $\lambda > 0$

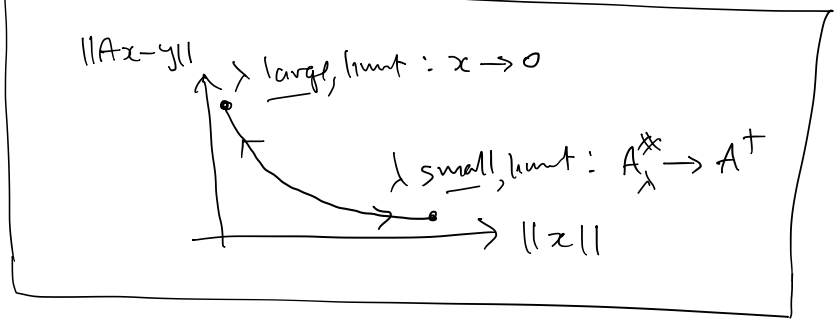
$$A_\lambda^* \to (A^TA)^{-1}A^T = A^+$$

ie unregularised limit is generalised inverse!

2. As $\lambda \to \infty$

$$A_\lambda^* \to \frac{1}{\lambda}A^T \to 0$$

so $x = A_\lambda^* y \to 0$ Shrunk soln to zero



Resolution:

$A: m \times n$

$A^{\#}: n \times m$

$$R_M^{Tik,\lambda} = \overset{\#}{A}_{\lambda} A \quad \} \overset{(1)}{\underset{(0)}{\rightleftarrows}}$$

$$R_D^{Tik,\lambda} = A \overset{\#}{A}_{\lambda} \quad \} \overset{(1)}{\underset{(2)}{\rightleftarrows}}$$

Note: $\overset{\#}{A}_{\lambda} \neq \hat{A}^{+}$ here

Recall $\tilde{A} = \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix}$

$$\hat{A}^{+} = (\tilde{A}^T \tilde{A})^{-1} \tilde{A}^T$$

$$= \left(\begin{bmatrix} A^T & \sqrt{\lambda} I \end{bmatrix} \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix} \right)^{-1} \begin{bmatrix} A^T & \sqrt{\lambda} I \end{bmatrix}$$

$$= (A^T A + \lambda I)^{-1} \begin{bmatrix} A^T & \sqrt{\lambda} I \end{bmatrix}$$

$$= \begin{bmatrix} \overset{\#}{A} & (A^T A + \lambda I)^{-1} \sqrt{\lambda} I \end{bmatrix} \} \; n \times (m+n)$$

$\overset{\#}{A}$: Tikhonov acts on y, $n \times m$

$(A^T A + \lambda I)^{-1} \sqrt{\lambda} I$: gets multiplied by zero, $n \times n$

Tikhonov regularisation in general

The Tikhonov form is

$$\boxed{\min \|Ax - y\|^2 + \lambda \|x\|^2}$$

→ zeroth order.

More general Tikhonov

$$\boxed{\min \|Ax - y\|^2 + \lambda \|Cx - b\|^2}$$

where C: weighting matrix or differential operator or : etc. } "prior information or constraints on models"

Also: ✓ other norms for data & or model space (can mix & match)

✓ nonlinear models $A(x)$ ie $\|A(x) - y\|^2 + \lambda \|x\|^2$ etc

note.

Note: regularisation is essentially the same as making up (or incorporating) additional data!

Amounts to $\tilde{A}x = \tilde{y}$,

where:

$$\tilde{A} = \begin{bmatrix} A \\ \sqrt{\lambda} I \end{bmatrix}, \quad \tilde{y} = \begin{bmatrix} y \\ 0 \end{bmatrix}$$

ie combining: $Ax = y$ } measured data

& $\sqrt{\lambda} I x = 0$ } 'made up' data.

'additional data model' } ideally, influence → 0 as real data increases
→ direct measurement of x

Statistical interpretation (see later?):

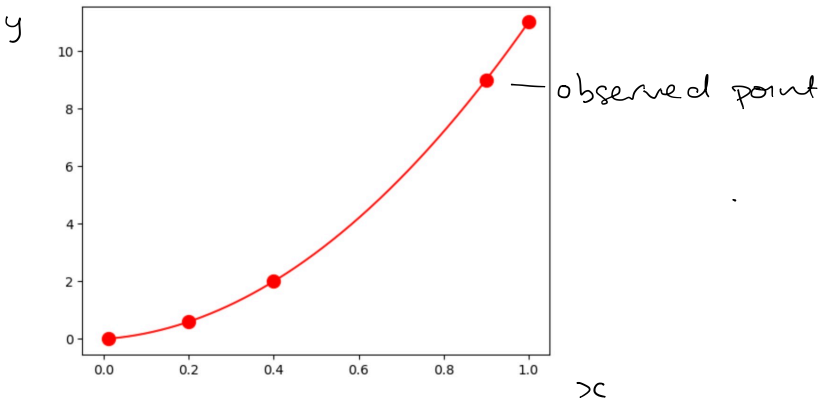
- For frequentist: synthesising data sets (eg product of two likelihoods) } 'mixed estimation'
- For Bayesians: combine likelihood & prior.

Important: both 'schools' can do!

Tikhonov Example: polynomial fitting

$y = x + 10x^2$ } True model.

5 observations:



Goal: fit the model:

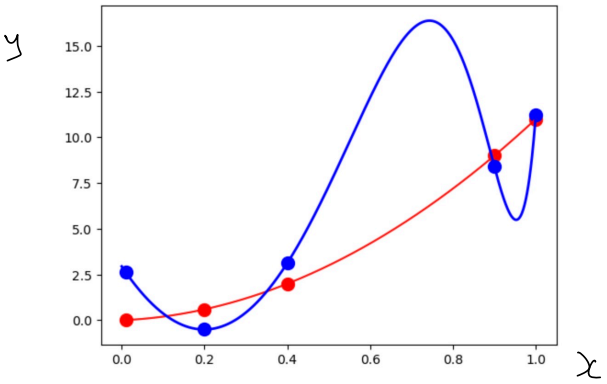
$$y = a_0 + a_1 x + \cdots a_9 x^{20}$$

parameters > observations!

→ will fit any noise too

Tikhonov Example : polynomial fitting

A^+ no regularisation:



Red: true

Blue: noisy

Note: exactly fitting noise

→ will get very different solution if observed again
↳ fit new noise etc

→ 'unstable'

Solution: consider regularisation!

Trade-off curve

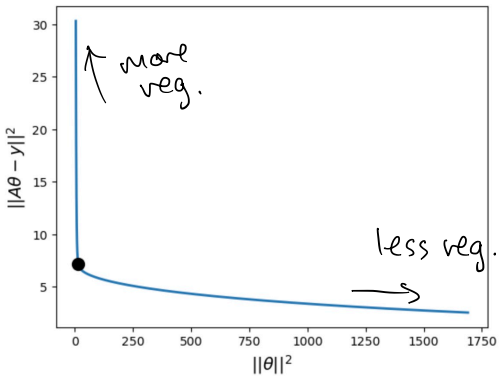
- Generate a grid of λ values
  (typically log-spaced, since 'asymptotic')

- Solve using ordinary least squares for augmented system $\tilde{A}\theta = \tilde{y}$

  → I just used $\theta = \tilde{A}^+ \tilde{y}$ via numpy's pseudo-inverse
  ↳ should use reg least squares

  → For nonlinear, need to solve with iterative etc method (see later)

Get:



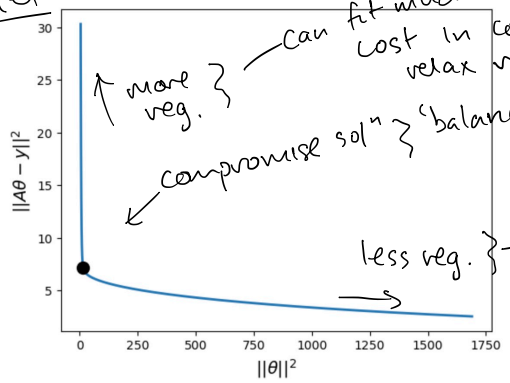
'Choosing' regularisation parameters (part I)

- The entire Pareto curve represents more info than any one solution
  → family of solutions as $f(\lambda)$
- But often want one 'good' solution.

→ Need to choose a single λ for these cases.

→ Simple idea: choose good 'compromise' soln:

Get:



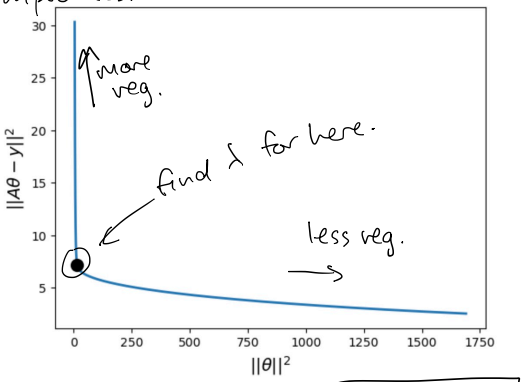
can fit much better for little cost in complexity if relax reg.

'balance', 'equal' cost

can make much simpler for little cost in fit, by inc. reg.

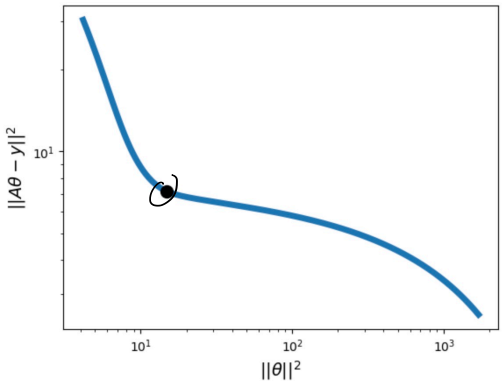
'Choosing' regularisation parameters

Compromise



On log-log plot, called 'L-curve' since often looks (more) L-shaped

↳ find 'elbow':



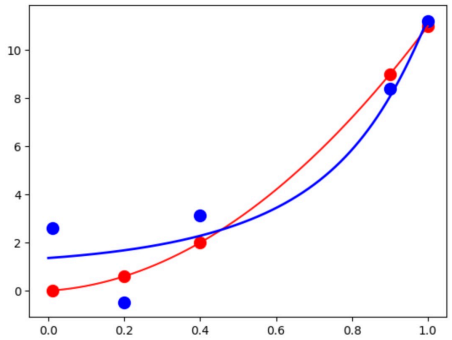
(Here isn't much more L-shaped ....)

→ see Aster et al or Hansen for better ex.)

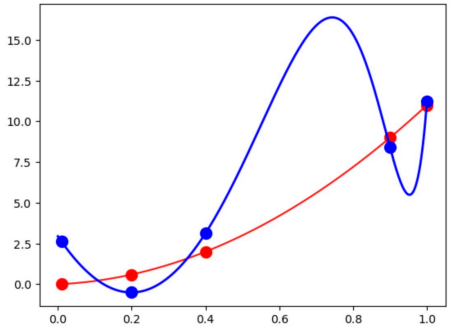
Example regularised soln

(λ chosen from curve 'by eye' here)

Regularised:



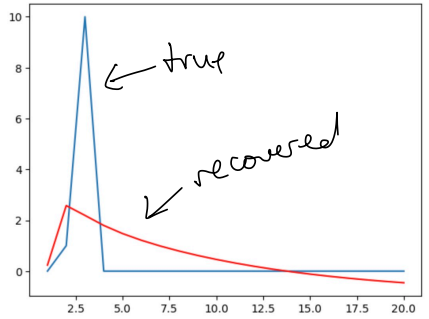
Compare to unregularised:



Note: in both cases we are fitting an order 20 polynomial to 5 data points!

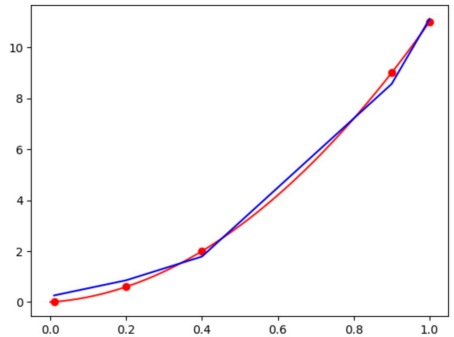
Resolution tests:

True model: $R_{\mathrm{Ty}}^{\mathrm{Tik},\lambda}(\theta_{\mathrm{true}})$



Note: regularisation introduces parameter biases

Ideal data: $R_{\mathrm{D}}^{\mathrm{Tik},\lambda}(A\theta_{\mathrm{true}})$ — noise-free data



Smoothes noisy data (see prev.) but preserves $A\theta_{\mathrm{true}}$ 'trend' well

Note: many ways to regularise!

Code snippets (see canvas for notebook)

```
def fmap_poly(tobs,n):
    A = np.zeros((len(tobs),n))
    for i, ti in enumerate(tobs):
        #A[i,:] = np.array([1,10*ti,-0.5*ti**2])
        A[i,:] = np.array([ti**j for j in range(0,n)])
    return A
```

```
#fine time grid
t = np.linspace(0,1,1000)

#observation times
tobs = np.array([0.01,0.2,0.4,0.9,1.0]) #over-determined

order = 20
A_poly_obs = fmap_poly(tobs,n=order)
A_poly_fine = fmap_poly(t,n=order)
```

```
#parameters
thetap_true = np.zeros(order)
thetap_true[1] = 1
thetap_true[2] = 10
thetap_true
```

```
#observed data
yp_obs = np.dot(A_poly_obs,thetap_true) + np.random.normal(0,1.5,size=len(tobs)) #with noise
```

Loop

```
alphav = np.logspace(-1.5, 0.5, num=1000)
```

9] ✓ 0.0s

```
thetas = np.zeros((len(alphav),len(thetap_true)))
theta_norms = np.zeros(len(alphav))
data_norms = np.zeros(len(alphav))
I = np.eye(len(thetap_true))
#I[0,0] = 0

for i, alphai in enumerate(alphav):

    #augment
    A_poly_aug = np.vstack([A_poly_obs,alphai*I])

    #invert
    A_poly_aug_pinv = np.linalg.pinv(A_poly_aug)
    thetas[i,:] = np.dot(A_poly_aug_pinv,yp_aug)

    #calc norms
    theta_norms[i] = np.linalg.norm(thetas[i,:],2)
    data_norms[i] = np.linalg.norm(yp_obs - np.dot(A_poly_obs,thetas[i,:]))
```

0] ✓ 0.0s