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 5: Matrix Calculus

Topics:

- Motivation
- Three key rules for derivatives
- First principles
- Key rules for differentials
- Examples

Eng Sci 721 : Lecture 5: Matrix Calculus

- Motivation
- Three key rules: derivatives for scalar & vector-valued functions of a vector
- First principles: derivatives & differentials for functions between any vector spaces (eg matrices $\xrightarrow{f}$ matrices)
- Other key rules: differentials
- Examples

Motivation:

Many inverse/learning from data/ML problems require derivatives (modern ML = 'differentiable programming')

of scalar/vector/matrix functions of scalar/vector/matrix inputs

Recall

1. Tall/overdetermined system: approximate solution

→ Define $r = y - Ax$ } residual 'error'

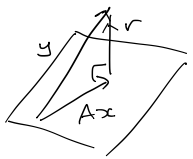
→ Measure size of error with a norm $\|\cdot\|$ (see handout for diff. types)

→ Typically assume $\|\cdot\|_2$

New problem:

minimise $\|y - Ax\|$, A & y given
$x \in \mathbb{R}^n$

- "best approximation"
- "closest approx"
- etc.



Minimiser of $\|\ \|$ & minimiser of $\|\ \|_2^2$ are same ($x \mapsto x^2$ is monotonic for $x \geq 0$)

⇒ Least squares approximation:

minimise $\|y - Ax\|_2^2$ (equiv. problem)
x

First:

$\|y - Ax\|^2$ as a matrix-vector expression

As: $\|z\|^2 = z^T z$

we have:

$$\|y - Ax\|^2 = (y - Ax)^T (y - Ax)$$

$$= (y^T - x^T A^T)(y - Ax)$$

Since:

$(A + B)^T = A^T + B^T$ & $(AB)^T = B^T A^T$ for matrices A, B

Also

$(A+B)(C+D) = AC + AD + BC + BD$

So $(y^T - x^T A^T)(y - Ax)$

$= y^T y - y^T Ax - x^T A^T y + x^T A^T Ax$

Also $y^T A x$ = scalar & scalar^T = scalar

& $(y^T A x)^T = x^T A^T y$

$= y^T A x$

So:

$$\|y - Ax\|^2 = y^T y - 2y^T Ax + x^T A^T A x$$

How to minimise wrt x?

→ take derivative wrt x &
set $= 0$ (actually just a necessary condition but ignore...)

→ Need to be able to differentiate matrix expressions with respect to vectors etc

→ Also often want to differentiate eg matrix expressions wrt matrices! (or tensors etc ~)

Differentiating vectors, matrices, tensors wrt vectors, matrices, tensors - - -

- important for inverse prob. & machine learning etc but... requires:

Matrix calculus, tensor calculus etc

→ multiple conventions / notation

(see eg Wiki page on Matrix Calculus)

- I will give you three key rules you can use for many problems, instead of remembering the details!

- I will also show you how to derive these & more from first principles

- The approach is based on 'matrix calculus' rather than eg tensor calculus, tho is equiv.

↳ we work in 'flattened' (vec) linear world!

ie vec (M_{ij}) vs M_{ij} etc.

First: derivatives of scalar & vector-valued functions of a vector

Eg: $f(x) = x^T A x$

x: vector input; $f(x)$: scalar output

Conventions

-> I'll use $D_x f$ for derivative of f wrt x, regardless of whether f, x are scalar, vector etc.

-> use 'Jacobian layout' eg vector f, x, components:

$$[D_x f]_{ij} = \frac{\partial f_i}{\partial x_j} = \begin{matrix} f_1 \\ \vdots \\ f_m \end{matrix} \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \quad \begin{matrix} x_1 & \cdots & x_n \end{matrix}$$

Note, if f is scalar-valued, this implies $D_x f$ is a row vector:

$$D_x f = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix}$$

Hence this is sometimes written

$$D_x f = \frac{\partial f}{\partial x^T}$$

to indicate the 'layout' direction is 'row like' in x.

We can define the gradient of f as

$$\nabla_x f = (D_x f)^T = \frac{\partial f^T}{\partial x}$$

(often drop transpose on f)

which again indicates how to 'lay out' results.

[Note $\left(\frac{\partial f}{\partial x^T}\right)^T = \frac{\partial f^T}{\partial x}$]

3 Key rules : (A, a independent of x)

Derivatives

Constant

$$D_x(a) = \frac{\partial}{\partial x^T}(a) = 0^T$$

Linear

$$D_x(Ax) = \frac{\partial}{\partial x^T}(Ax) = A$$

Quadratic

$$D_x(x^T A x) = \frac{\partial}{\partial x^T}(x^T A x)$$

$$= x^T(A + A^T)$$

Know these !

[We can also show eg

- Derivative of scalar, possibly dep. on x :

$$D_x(a(x)) = D_x(a) = D_x(a^T)$$

- (Multivariable vector) chain rule

$$D_x(f(h(x), g(x)) = \frac{\partial}{\partial x^T} f(h, g)$$

$$= (D_g f)\cdot(D_x g) + (D_h f)\cdot(D_x h)$$

$$= \frac{\partial f}{\partial g^T}\frac{\partial g}{\partial x^T} + \frac{\partial f}{\partial h^T}\frac{\partial h}{\partial x^T}$$

Imply eg 'product rule' of form :

$$D_x(h^T g) = h^T D_x g + g^T D_x h$$

$$= h^T \frac{\partial g}{\partial x^T} + g^T \frac{\partial h}{\partial x^T}$$

]

where do these come from??

→ soon!

→ first, consider ⟶

Back to least squares!

$$\min_x f(x) = y^T y - 2y^T A x + x^T A^T A x$$

⇒ set $D_x f = 0^T$ (1) (row vector)

(or $\nabla_x f = 0$ ...)

3 Rules:

$$D_x (y^T y) = 0^T$$

$$D_x (-2y^T A x) = -2y^T A$$

$$D_x (x^T A^T A x) = x^T (A^T A + (A^T A)^T)$$

$$= 2x^T A^T A$$

(1) ⇒ $-2y^T A + 2x^T A^T A = 0^T$

⇒ $-A^T y + A^T A x = 0$

⇒ $\boxed{A^T A x = A^T y}$ as before!

Recall:

Least squares approximation

We have seen this leads to....

The normal equations: [normal? $A^T(Ax-y)=0$ geometric]

$$A^TAx = A^Ty$$

→ (if we assume the n cols of A are linearly independent then A^TA is invertible (see handout) & so get unique approximate soln

$$x^* = (A^TA)^{-1}A^Ty$$

First principles matrix calculus

- Derivatives, linear approximation & differentials

[This approach is heavily influenced by that of Magnus & Neudecker, ⋆ A lot beyond scope! ⋆]

What is a derivative?

Recall the usual def^n for $f:\mathbb{R}\to\mathbb{R}$ at x_0:

(1) $$\frac{df}{dx}(x_0) := \lim_{h\to 0} \frac{f(x_0+h)-f(x_0)}{h}$$

This is equivalent to, for some $a\in\mathbb{R}$

(2) $$f(x_0+h) = f(x_0) + ah + o(h), h\to 0$$

where '$o(h)$' is 'little oh' & means, roughly, 'goes to zero faster than h,'

& we define: $\frac{df}{dx}(x_0) := a$ → from the above.

(a depends on x_0).

Linear approximation & differentials

The form:

$$f(x_0+h) = f(x_0) + ah + o(h),\ h\to 0$$

emphasises that f can be approximated by the linear function ah at x_0.

[note: ah should really be $a(x_0)\cdot h$ to allow different a at each point]

We further define the differential of f to be the linear part at x_0:

$$\boxed{df = a\cdot h = \frac{df}{dx}\cdot h} \quad \text{via } \frac{df}{dx} \overset{\text{defn}}{:=} a$$

ie, in full detail..

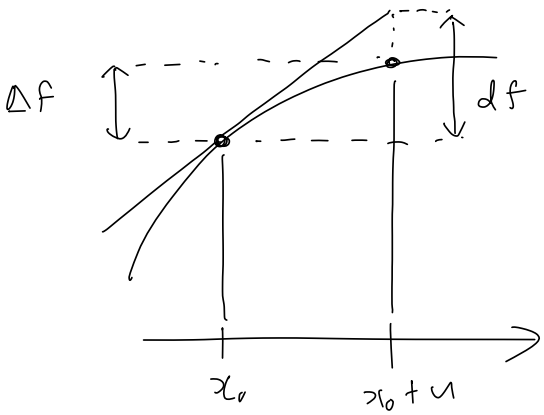
$$\boxed{df(x_0;h) := a(x_0)\cdot h = \frac{df}{dx}(x_0)\,h}$$

We now allow this linear approximation to be evaluated for any 'increment', not just $h\to 0$, ie

$$\boxed{df(x_0;u) = a(x_0)\cdot u} \text{ for any } u. \quad (u \text{ eg finite})$$

$$\boxed{= \frac{df}{dx}(x_0)\cdot u} \text{ by defn.}$$

Geometric picture



$$\Delta f = f(x_0+u) - f(x_0) = \text{actual change in } f \text{ for increment } u$$

$$df = a(x_0)\cdot u = \frac{df}{dx}(x_0)\cdot u \qquad (u \text{ not necessarily} \to 0)$$

for $a(x_0)$ satisfying

$$\Delta f = a(x_0)\cdot h + o(h),\ \forall h\to 0$$

Generalisations to higher-dim

Both defns (1) & (2) allow us to generalise derivatives & differentials to functions having vector inputs & vector outputs

[Here we mean 'vectors' in the general sense, so can apply to eg 'matrices as vectors'!]

These generalisations make different assumptions

(1) → Gâteaux derivative & differential
↳ weak, exists more often

(2) → Fréchet derivative & differential
↳ stronger, can have (1) but not (2)

For us, both exist & are equal tho!

(1) **Frechet** (linear approx., implicit)

If $f(x_0+h) = f(x_0) + Ah + o(\|h\|)$

where $f: \mathbb{R}^n \to \mathbb{R}^m$ $\quad \forall h \in \mathbb{R}^n \to 0 \in \mathbb{R}^n$

$x_0, h \in \mathbb{R}^n$

$A \in \mathbb{R}^{m \times n}$ ie matrix

then this **defines** ! (implicitly !)

- derivative: $Df(x_0) = A$
- differential: $df(x_0, u) = Df(x_0)\,u$, $u \in \mathbb{R}^n$
  - ↳ also write $df(x_0) = Df(x_0)\,dx$ ie $u = dx$
  - or just $df = Df\,dx$

→

(2) **Gateaux** (directional derivative, explicit)

If $\lim_{t\to 0} \frac{f(x_0+th)-f(x_0)}{t} = Ah, \ \forall h\in\mathbb{R}^n$

where $f:\mathbb{R}^n\to\mathbb{R}^m$
$t\in\mathbb{R}$ (scalar)
$x_0, h\in\mathbb{R}^n$
$A\in\mathbb{R}^{m\times n}$ ie matrix

Then this **defines** (by 'direct calculation')

- derivative: $Df(x_0) = A$
- differential: $df(x_0;u) = Df(x_0)\,u$, $u\in\mathbb{R}^n$

↳ write $df(x_0) = Df(x_0)\,dx$ ie $u=dx$

or just $df = Df\,dx$

Subtleties

- Gateaux differentiability allows the defn of partial derivatives:

eg $df(x_0;e_j) = \lim_{t\to 0} \frac{f(x_0+te_j)-f(x_0)}{t}$

[where $e_j = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \leftarrow j$]

$= Ae_j$ (if exists)

$=$ j^{th} col of A

If $f_i = i^{th}$ component of f,

$df_i(x_0;e_j) = \lim_{t\to 0} \frac{f_i(x_0+te_j)-f_i(x_0)}{t}$

$=$ i^{th} row, j^{th} col of A

$= e_i\, df(x_0;e_j)$

→ defines $D_j f_i(x_0) = \frac{\partial f_i}{\partial x_j}(x_0)$ note order

& $Df(x_0) = \underbrace{[D_j f_i(x_0)]}_{\text{array of partial deriv.}}{}_{i=1:n,\, j=1:m}$

$=$ **Jacobian**

Subtleties

- We can have Gateaux differentiability / existence of partial derivatives / Jacobian matrix but no Frechet derivative (linear approx.)
  → we will ignore these cases!
  → take 'differentiable' to mean 'Frechet exists' (& equals Gateaux)

Leads to correspondence (identification thm)

If $df(x_0; u) = A(x_0)\,u$ for some matrix A depending on x_0

Then $A(x_0) = \left[D_j f_i(x_0) \right] = Df(x_0)$ (partial derivatives; Jacobian)

& conversely

(Derivative is uniquely determined via linearisation or partial derivative / Jacobian matrix)

ie

→ * | If $df = A\,dx$ then $Df = A$ | * ←

(& vice-versa)

Derivatives & differentials of matrix-valued functions of matrices

Recall that matrices can be directly thought of as 'abstract' vectors in the vector-space $\mathbb{R}^{m \times n}$

eg $M_3 = aM_1 + bM_2$ etc

→ these are not column vectors in the usual sense

→ However, we can also explicitly map these to column vectors & use this familiar space for manipulations:

$M \to \mathrm{vec}(M)$

'vec' is linear so:

$$\mathrm{vec}(aM_1 + bM_2) = a\,\mathrm{vec}(M_1) + b\,\mathrm{vec}(M_2)$$

etc.

→

Implications for derivatives

If $f : \mathbb{R}^{m \times n} \to \mathbb{R}^{m \times n}$

is differentiable in the Frechet sense, we have a linear approx. in the vector space of matrices:

$$\boxed{df(M_0, dM) = \mathcal{L}(M_0)\, dM}$$

where $\mathcal{L}$ is a 'linear operator' mapping matrices to matrices.

Importantly, if M is $m \times n$ then $\boxed{dM \text{ is also } m \times n}$

However $\boxed{\mathcal{L} \text{ is not a matrix}}$!

→ It is a 'tensor', here represented by a 4D array of dimensions $m \times n \times m \times n$

→

Tensors in brief. Much more to it!

→ Represented by multidimensional arrays (0, 1, 2, 3 etc dimensional)

$A_{ijkl\ldots}$

→ Group 'axes' into two sets: inputs & outputs

eg $ijkl \to (i,j|k,l)$ or A^{ij}_{kl}

→ write or just A_{ijkl} if clear

→ Represent linear mappings between tensors of dimensions of 'input' & dimensions of 'output'

eg A^{ij}_{kl} (output) maps 2D arrays to 2D arrays (input) & is a 4D array

'Type': (2, 2) (output dim, input dim)

→ we can re-group indices to change input-output specification:

A^i_j : maps vector x^j to vector y^i

→ can also 'input from other side'

eg $x_i \to A^i_j \to y_j$

Derivatives as tensors

So ... if $f: \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$

matrices $m\times n$ — matrices $m\times n$

Then $df = \mathcal{L}\, dM$

$\mathcal{L}$ is a 4D tensor!

components $\mathcal{L}^{ij}_{kl}$

Formula for 'applying' $\mathcal{L}$ to B to get A:

$$A^{ij} = \sum_{kl} \mathcal{L}^{ij}_{kl} B^{kl}$$

$$= \mathcal{L}^{ij}_{kl} B^{kl}$$ (implicit sum: Einstein convention)

→ OK to do this ... but can also 'flatten'!

→

Derivative matrices (cf tensor)

We can also define the derivative (matrix) eg a matrix function of a matrix variable

→ we simply map matrices to col. vectors using vec!

- Derivative is linear approx. to vector → vector mapping } matrix

→ Define for $F(X): \mathbb{R}^{m\times n} \to \mathbb{R}^{p\times q}$

$$d\,\mathrm{vec}(F(X)) = A(X)\, d\,\mathrm{vec}(X)$$

$$\Leftrightarrow$$

$$D_X F(X) = A(X)$$

$$\Leftrightarrow$$

$$\frac{\partial\, \mathrm{vec}\, F(X)}{\partial (\mathrm{vec}\, X)^T} = A(X)$$

ie $D_X F$ represents the derivative matrix (vectorised)

Differentials vs derivatives

In the original space, dM is same shape as M, while 'DM' becomes a higher-order tensor unless vectorised

→ It is often easier to do some calculations with differentials in the original space, & then, if needed, vectorise to get a derivative matrix

This is helped by the following useful rules that apply to differentials directly in their natural space (in contrast, derivative expressions can become complex)

Key rules of differentials (original space)

For matrix functions F, G, constant matrix A, constant number α,

- $dA = 0$
- $d(\alpha A) = \alpha\, dA$
- $d(A+B) = dA + dB$
- $d(A-B) = dA - dB$
- $d(UV) = (dU)V + U(dV)$
- $d(U^T) = (dU)^T$
- $d(\mathrm{vec}(U)) = \mathrm{vec}(dU)$
- $d(\text{linear function}(U)) = \text{linear f}(dU)$
- Also, if $H = G(F(X))$

$$dH(X_0; dX) = dG\left(F(X_0); dF(X_0, dX)\right)$$

(chain rule / Cauchy invariance)

Examples

- Consider $f(x) = x^T A x$
  - → quadratic (nonlinear)
  - $f: \mathbb{R}^n \to \mathbb{R}$ } already 'vectorised'
    - vector
    - scalar

$$df = df(x; dx)$$
$$= (dx)^T Ax + x^T A\, dx$$
$$= x^T A^T dx + x^T A\, dx$$
$$= x^T (A^T + A)\, dx$$

$$\Rightarrow \boxed{Df(x) = x^T (A^T + A)} \checkmark$$

Examples (harder!)

matrix-val. f of matrix.

- Suppose A is invertible. $A^{-1} = f(A)$

Then $A^{-1}A = I$

$$d(A^{-1}A) = dI = 0$$

$$\Rightarrow d(A^{-1})A + A^{-1}dA = 0$$
$$d(A^{-1})AA^{-1} + A^{-1}dAA^{-1} = 0$$

$$\Rightarrow \boxed{d(A^{-1}) = -A^{-1}dA\,A^{-1}}$$

$$\text{vec}(d(A^{-1})) = -(A^T)^{-1} \otimes A^{-1}\, \text{vec}(dA)$$

$$\Rightarrow \boxed{D_A A^{-1} = -(A^T)^{-1} \otimes A^{-1}} \quad !$$

✓ (Magnus & Neudecker)

Examples (harder!)

$x \in \mathbb{R}^n$

- Consider $F(x) = xx^T$

F : vector → matrix

$$dF = (dx)x^T + x\,d(x^T)$$
$$= (dx)x^T + x(dx)^T$$

$$\text{vec}(dF) = \text{vec}\left(I_n(dx)x^T + x(dx)^T I_n\right)$$
$$= (x \otimes I)\,\text{vec}(dx) + (I \otimes x)\,\text{vec}(dx^T)$$
$$= (x \otimes I)\,dx + (I \otimes x)\,dx$$
$$= \left[x \otimes I + I \otimes x\right]\overbrace{\text{vec}(dx)}^{dx}$$

$$\Rightarrow \quad D_x F = x \otimes I + I \otimes x \quad !$$

✓ (see Magnus & Neudecker)

Higher derivatives: second order

Suppose $\phi : \mathbb{R}^n \to \mathbb{R}$ (scalar-valued)

- Differentials:

$$\rightarrow \boxed{d^2\phi := d(d\phi)} \leftarrow$$

- Derivatives:

Define Hessian matrix as

$$\boxed{H\phi = \nabla(D\phi)^T}$$
$$\boxed{= \frac{\partial^2 \phi}{\partial x\, \partial x^T}}$$

ie $\boxed{[H\phi]_{ij} = D_{ij}\phi}$

- Relation (identification):

If $\boxed{d^2\phi = (dx)^T B\, dx}$ for some B,

Then $\boxed{H\phi = \frac{B + B^T}{2}}$ } note! related to symmetry of Hessian

& conversely.

Exercises

- Re-derive the 'normal equations' for least squares from memory
- Derive the 3 key rules for derivatives from rules for differentials
- If $\phi = x^T A x$

  find $d(d\phi)$
- Look up automatic differentiation in ML libraries
  - Tensorflow
  - pytorch
  - jax
  - ⋮
  - etc !