

Lecture 5: Regularisation, Multivariate Models, and Nonlinear Problems

Attendance code: K8PE3UZF

Iain Styles

25 October 2019

Learning Outcomes

By the end of the this lecture you should be able to

- ▶ Understand the effect of regularisation on regression problems
- ▶ Be able to formulate and solve linear regression problems with multiple independent and dependent variables
- ▶ Appreciate how non-linear problems are different, and how they can be approached.

Recap

- ▶ Uniform prior on model parameters \mapsto $L_{LS} + \lambda w^T w$ Least-squares loss
- ▶ Gaussian prior on model parameters \mapsto least squares loss with L_2 penalty term
- ▶ Encourages model parameters to not grow too large
- ▶ Helps to reduce overfitting
- ▶ How do we apply it?

Regularised Loss Functions

- ▶ Generalised regularised loss function

$$\mathcal{L}(\mathbf{w}) = \mathcal{L}_{\text{err}}(\mathbf{w}) + \lambda R(\mathbf{w}) \quad (1)$$

- ▶ Model-data mismatch plus “penalty” term
- ▶ For the specific case of LSE + ℓ_2 penalty (Gaussian prior)

$$\mathcal{L}(\mathbf{w}) = (\mathbf{y} - \Phi\mathbf{w})^T (\mathbf{y} - \Phi\mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w} \quad (2)$$

- ▶ Minimising loss requires both terms to be minimised
- ▶ λ controls the width of the Gaussian prior and hence the balance between model fitting and parameter shrinkage.

Solving Regularised Least Squares

- ▶ One reason L_2 is common is its closed-form analytic solution
- ▶ Differentiate loss function with respect to \mathbf{w} and set to zero to minimise

$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= (\mathbf{y} - \Phi\mathbf{w})^T (\mathbf{y} - \Phi\mathbf{w}) + \lambda\mathbf{w}^T\mathbf{w} \\ \frac{\partial \mathcal{L}_{\text{LSE}}(\mathbf{w})}{\partial \mathbf{w}} &= -2\Phi^T (\mathbf{y} - \Phi\mathbf{w}) + 2\lambda\mathbf{w}\end{aligned}$$

- ▶ Set to zero to minimise

$$\Phi^T \mathbf{y} - (\Phi^T \Phi - \lambda \mathbf{I}) \mathbf{w}^* = 0$$

identity matrix

- ▶ Modified normal equations, can be solved in the same way

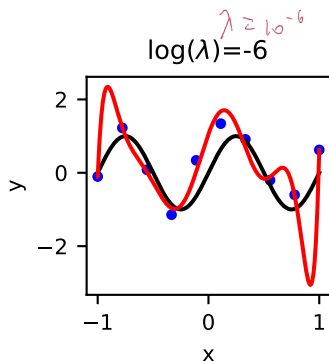
Terminology

Known by a number of names:

- ▶ *Ridge regression*.
- ▶ *L_2 regularisation*, because $\sum_i w_i^2$ is the L_2 norm of \mathbf{w} , written as $\|\mathbf{w}\|_2^2$.
- ▶ *Weight decay*, because it pushes weights towards zero.
- ▶ *Tikhonov regularisation*, of which it is a special case.
Tikhonov regularisation uses $R(\mathbf{p}) = \|\mathbf{\Gamma}\mathbf{w}\|$. Here, we have $\mathbf{\Gamma} = \lambda\mathbf{I}$.

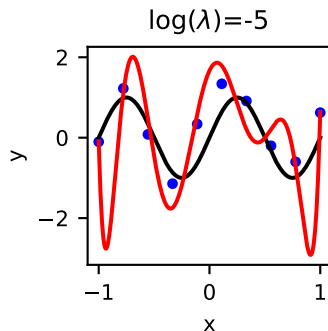
Regularisation in Practice

- ▶ L_2 regression used to prevent overfitting
- ▶ Prevents model weight growing large to fit noise.
- ▶ Large values of λ “smooth” fluctuations



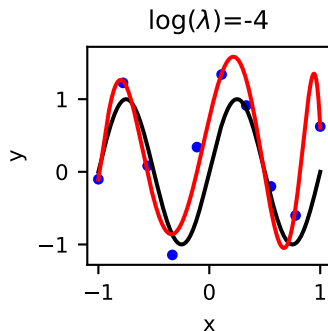
Regularisation in Practice

- ▶ L_2 regression used to prevent overfitting
- ▶ Prevents model weight growing large to fit noise.
- ▶ Large values of λ “smooth” fluctuations



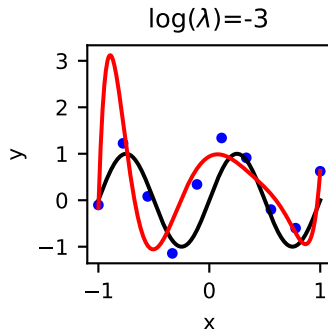
Regularisation in Practice

- ▶ L_2 regression used to prevent overfitting
- ▶ Prevents model weight growing large to fit noise.
- ▶ Large values of λ “smooth” fluctuations



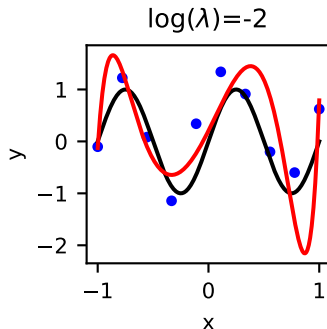
Regularisation in Practice

- ▶ L_2 regression used to prevent overfitting
- ▶ Prevents model weight growing large to fit noise.
- ▶ Large values of λ “smooth” fluctuations



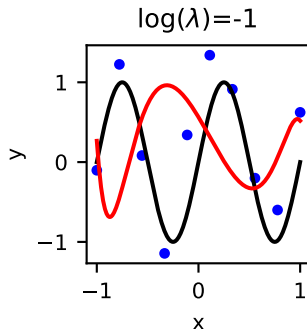
Regularisation in Practice

- ▶ L_2 regression used to prevent overfitting
- ▶ Prevents model weight growing large to fit noise.
- ▶ Large values of λ “smooth” fluctuations



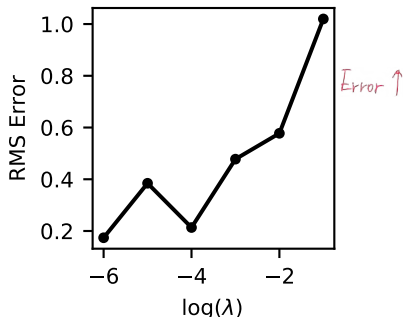
Regularisation in Practice

- ▶ L_2 regression used to prevent overfitting
- ▶ Prevents model weight growing large to fit noise.
- ▶ Large values of λ “smooth” fluctuations



Effect of Regularisation

- ▶ Regularisation increases training error but improves generalisation
- ▶ Penalty term reduce model weights



$\log_{10} \lambda$	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9
-6	1.06	8.31	-17.92	-76.20	76.16	250.58	-112.92	-350.94	53.88	168.61
-5	1.67	5.77	-41.44	-29.12	212.84	31.81	-356.24	1.30	183.44	-9.40
-4	0.74	6.51	-5.21	-33.75	1.46	33.96	20.81	13.89	-17.55	-20.25
-3	0.94	1.29	-9.30	5.42	15.29	-21.02	5.50	-4.36	-12.20	19.08
-2	0.20	4.26	2.26	-10.00	-5.63	-4.88	-0.83	2.60	4.33	8.47
-1	0.56	-2.12	-1.35	4.00	-0.63	1.44	0.50	-0.84	1.30	-2.35

Limitations

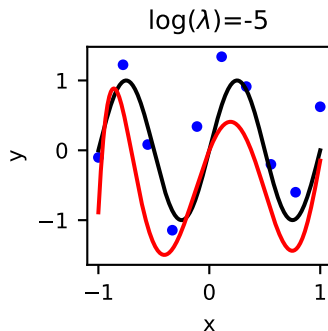
- ▶ L_2 regularisation tends to “smooth” the fit
- ▶ Popular for image denoising
- ▶ But does not work well for data that really does have “fast” fluctuations.
- ▶ Other choices can overcome this
- ▶ Most general form: $R(\mathbf{w}) = \sum_i |w_i|^p$ $\rightarrow \leq 1$ popular \uparrow
- ▶ $p = 1$ is very common - L_1 or *lasso* regularisation

L_1 Regularisation

- ▶ Also known as the Lasso methods
- ▶ $R(\mathbf{w}) = \sum_i |w_i|$
- ▶ Tends to promote *sparsity* in model parameters
- ▶ No closed form solution
- ▶ Turn to `scikit-learn` for implementation

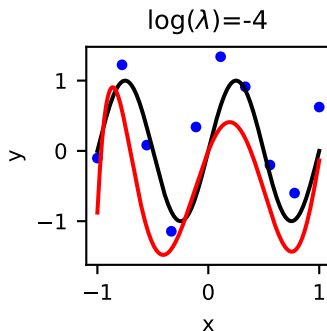
Lasso in Practice

- ▶ L_2 regression used to prevent overfitting
- ▶ Prevents model weight growing large to fit noise.
- ▶ Large values of λ “smooth” fluctuations



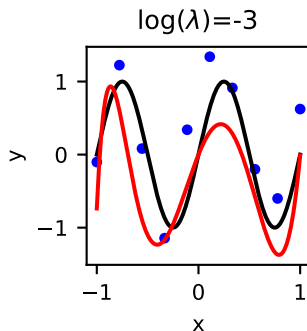
Lasso in Practice

- ▶ L_2 regression used to prevent overfitting
- ▶ Prevents model weight growing large to fit noise.
- ▶ Large values of λ “smooth” fluctuations



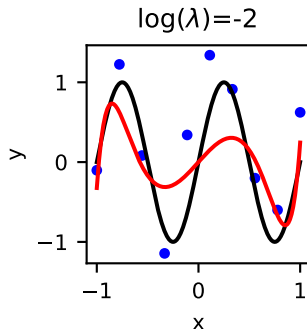
Lasso in Practice

- ▶ L_2 regression used to prevent overfitting
- ▶ Prevents model weight growing large to fit noise.
- ▶ Large values of λ “smooth” fluctuations



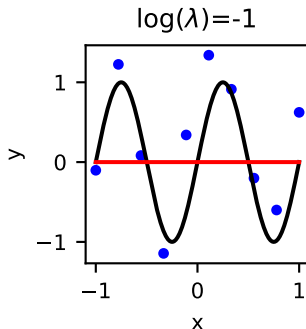
Lasso in Practice

- ▶ L_2 regression used to prevent overfitting
- ▶ Prevents model weight growing large to fit noise.
- ▶ Large values of λ “smooth” fluctuations



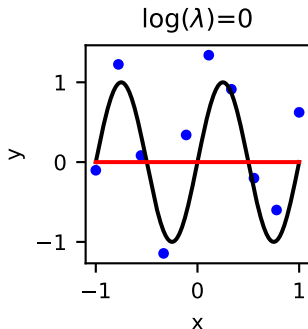
Lasso in Practice

- ▶ L_2 regression used to prevent overfitting
- ▶ Prevents model weight growing large to fit noise.
- ▶ Large values of λ “smooth” fluctuations



Lasso in Practice

- ▶ L_2 regression used to prevent overfitting
- ▶ Prevents model weight growing large to fit noise.
- ▶ Large values of λ “smooth” fluctuations



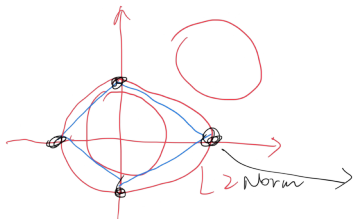
L_1 -regularised model parameters

$\log_{10} \lambda$	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9
-5	0.00	3.59	-0.00	-15.72	0.00	11.78	0.00	1.96	0.00	-1.60
-4	0.00	3.54	0.00	-15.27	0.00	11.05	0.00	1.93	0.00	-1.24
-3	0.00	3.10	-0.00	-12.02	-0.00	5.44	-0.00	3.49	-0.00	0.00
-2	0.00	0.95	-0.00	-3.76	-0.00	-0.00	-0.00	0.00	-0.00	2.74
-1	0.00	-0.05	0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00
0	0.00	-0.00	0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00

- Progressive **sparsification** of model parameters

Regularisation in Practice

- ▶ The correct regulariser and choice of λ is very problem dependent
- ▶ Smooth or sparse solution?
- ▶ Rigorous cross-validation often needed to selection λ
- ▶ Enables complex models to be used and then controlled



$$L_1 = \sum_i |w_i| \quad \text{Norm}$$

sparse \uparrow because one of the dimension is 0.

Regularisation in Practice

- ▶ The correct regulariser and choice of λ is very problem dependent
- ▶ Smooth or sparse solution?
- ▶ Rigorous cross-validation often needed to selection λ
- ▶ Enables complex models to be used and then controlled
- ▶ And now a change of topic. . .

Multivariate Regression

- ▶ One independent variable x , one dependent variable y
- ▶ How to deal with multiple variables of each type?
- ▶ Univariate methods generalise quite straightforwardly
- ▶ Consider two independent variables, and a model that contains a constant term and linear terms in the two variables:

$$y = w_0 + w_1 x_1 + w_2 x_2$$

independent

Multivariate Basis Matrix

- ▶ Same form as polynomial expansion
- ▶ Basis matrix has one row per sample, one column per feature

$$\Phi = \begin{pmatrix} 1 & x_{1,1} & x_{2,1} \\ 1 & x_{1,2} & x_{2,2} \\ \vdots & \vdots & \vdots \\ 1 & x_{1,N} & x_{2,N} \end{pmatrix}$$

Multivariate Basis Matrix

- ▶ Same form as polynomial expansion
- ▶ Basis matrix has one row per sample, one column per feature

$$\Phi = \begin{pmatrix} 1 & x_{1,1} & x_{2,1} \\ 1 & x_{1,2} & x_{2,2} \\ \vdots & \vdots & \vdots \\ 1 & x_{1,N} & x_{2,N} \end{pmatrix}$$

- ▶ A more complex quadratic model is similar

$$y = w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2 + w_5x_1x_2$$

has basis matrix

$$\Phi = \begin{pmatrix} 1 & x_{1,1} & x_{2,1} & x_{1,1}^2 & x_{2,1}^2 & x_{1,1}x_{2,1} \\ 1 & x_{1,2} & x_{2,2} & x_{1,2}^2 & x_{2,2}^2 & x_{1,2}x_{2,2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{1,N} & x_{2,N} & x_{1,N}^2 & x_{2,N}^2 & x_{1,N}x_{2,N} \end{pmatrix}$$

Multiple Dependent Variables

- ▶ Assumes same set of features/basis for each output
- ▶ Approach 1: treat all outputs separately:

$$y_1(x_1, \dots, x_K) = \sum_j w_{1,j} \phi_j(x_1, \dots, x_K) \rightarrow \Phi^T \Phi \mathbf{w}_1 = \Phi^T \mathbf{y}_1$$

$$y_2(x_1, \dots, x_K) = \sum_j w_{2,j} \phi_j(x_1, \dots, x_K) \rightarrow \Phi^T \Phi \mathbf{w}_2 = \Phi^T \mathbf{y}_2$$

$$\vdots \quad \vdots$$
$$y_L(x_1, \dots, x_K) = \sum_j w_{L,j} \phi_j(x_1, \dots, x_K) \rightarrow \Phi^T \Phi \mathbf{w}_L = \Phi^T \mathbf{y}_L$$

where $\mathbf{y}_j = [y_{j,1}, y_{j,2}, \dots, y_{j,N}]^T$ and $\mathbf{w}_i = [w_{i,1}, w_{i,2}, \dots, w_{i,M}]$.

Multiple Dependent Variables

- Approach 2: "joint" optimisation over all of the dependent variables in a single step

$$\begin{pmatrix} y_{1,1} & y_{2,1} & \dots & y_{K,1} \\ y_{1,2} & y_{2,2} & \dots & y_{K,2} \\ \vdots & \vdots & \ddots & \vdots \\ y_{1,N} & y_{2,N} & \dots & y_{K,N} \end{pmatrix} = \begin{pmatrix} \phi_1(x_1) & \phi_2(x_1) & \dots & \phi_M(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \dots & \phi_M(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(x_N) & \phi_2(x_N) & \dots & \phi_M(x_N) \end{pmatrix} \times \begin{pmatrix} w_{1,1} & w_{2,1} & \dots & w_{K,1} \\ w_{1,2} & w_{2,2} & \dots & w_{K,2} \\ \vdots & \vdots & \ddots & \vdots \\ w_{1,M} & w_{2,M} & \dots & w_{K,M} \end{pmatrix}$$
$$\rightarrow \mathbf{Y} = \mathbf{\Phi} \mathbf{W}$$

- Joint solution via normal equations $\mathbf{\Phi}^T \mathbf{\Phi} \mathbf{W} = \mathbf{\Phi}^T \mathbf{Y}$

Non-linear Regression

- ▶ Non-examinable
- ▶ How to deal with nonlinear models, eg $y(x) = a \sin bx$, where we need to find a and b .
- ▶ Modelling function is not linear in unknowns
- ▶ Cannot minimise loss directly
- ▶ Instead, locally reduce the loss
- ▶ Do this iteratively to find global minimum

Locally linearise the loss

- ▶ Approximating loss as a straight line over a small region using *Taylor's Theorem*:

$$\begin{aligned}f(x_i, \mathbf{w} + \Delta \mathbf{w}) &\approx f(x_i, \mathbf{w}) + \sum_{j=1}^m \frac{\partial f(x_i)}{\partial w_j} \Delta w_j \\&= f(x_i, \mathbf{w}) + \sum_{j=1}^m J_{ij} \delta w_j\end{aligned}$$

where matrix \mathbf{J} has components $J_{ij} = \frac{\partial f(x_i)}{\partial w_j}$

- ▶ The **residual** becomes

$$\mathbf{r}(\mathbf{w} + \Delta \mathbf{w}) = \mathbf{y} - \mathbf{f}(\mathbf{w}) - \mathbf{J} \Delta \mathbf{w}$$

where $f_i(\mathbf{w}) = f(x_i, \mathbf{w})$. The loss is therefore

$$\mathcal{L}(\mathbf{w} + \Delta \mathbf{w}) = (\mathbf{y} - \mathbf{f}(\mathbf{w}) - \mathbf{J} \Delta \mathbf{w})^T (\mathbf{y} - \mathbf{f}(\mathbf{w}) - \mathbf{J} \Delta \mathbf{w}).$$

By minimising this quantity, we find the *change* in \mathbf{w} that gives us the largest reduction in the error.

Locally minimise the loss

- ▶ We differentiate as before and set to zero:

$$(\mathbf{J}^T \mathbf{J}) \Delta \mathbf{w}^* = \mathbf{J}^T (\mathbf{y} - \mathbf{f}(\mathbf{w}))$$

which we now how to solve

- ▶ It is common practice to make a small modification to this:

$$(\mathbf{J}^T \mathbf{J} + \lambda \text{diag}(\mathbf{J}^T \mathbf{J})) \Delta \mathbf{w}^* = \mathbf{J}^T (\mathbf{y} - \mathbf{f}(\mathbf{w}))$$

- ▶ Regularisation that penalises large updates to the parameter vector along directions with steep gradients, encouraging movement along shallow gradients and thus a faster convergence.

The Levenberg-Marquadt Algorithm

```

Data: Data  $x, y$ ; Initial
      parameter guess  $w_0$ ;
      Function  $f(x, w)$ ;
      Jacobian  $Jac(x, w)$ ;
      initial values of  $\lambda, \nu$ ;
Result:  $w, e(w)$ ; parameters
      that minimise the
      least squares error, the
      error.
% Evaluate the error at
% the initial parameter
% value
 $w \leftarrow w_0$ ;
 $e \leftarrow (y - f(x, w))^T (y - f(x, w))$ ;
% Iterate until
% convergence
Converged  $\leftarrow$  false;
while Converged == false do
    % Calculate the
    % Jacobian
     $J \leftarrow \frac{\partial f}{\partial w}$ ;
    % Calculate update for
    % different values of  $\lambda$ 
    % until we improve the
    % error
    LambdaSet  $\leftarrow$  false;
    while LambdaSet == false
        do
            % Calculate  $\Delta w$  for
             $\lambda = \lambda_0$  and  $\lambda = \lambda_0/\nu$ 
             $\Delta_1 \leftarrow$ 
             $(J^T J + \lambda \text{diag}(J^T J)) \setminus$ 
             $J^T (y - f(x, w))$ ;
             $\Delta_0 \leftarrow$ 
             $(J^T J + (\lambda/\nu) \text{diag}(J^T J)) \setminus$ 
             $J^T (y - f(x, w))$ ;
            % Calculate the
            % error at the new
            % parameter values
             $e_1 \leftarrow (y - f(x, w +$ 
             $\Delta_1))^T (y - f(x, w +$ 
             $\Delta_1))$ ;
             $e_0 \leftarrow (y - f(x, w +$ 
             $\Delta_0))^T (y - f(x, w +$ 
             $\Delta_0))$ ;
            if  $e_1 < e$  then
                % Set new values
                % of  $w$  and  $e$ ,
                % storing old
                % value as  $e'$ 
                 $w \leftarrow w + \Delta_1$ ;
                 $e' \leftarrow e$ ;
                 $e \leftarrow e_1$ ;
                % Current  $\lambda$  is OK
                LambdaSet  $\leftarrow$  true;
            else if  $e_0 < e$  then
                % Set new values
                % of  $w$  and  $e$ ,
                % storing old
                % value as  $e'$ 
                 $w \leftarrow w + \Delta_0$ ;
                 $e' \leftarrow e$ ;
                 $e \leftarrow e_0$ ;
                 $\lambda \leftarrow \lambda/\nu$ ;
                % Current  $\lambda$  is OK
                LambdaSet  $\leftarrow$  true;
            else
                 $\lambda \leftarrow \lambda \times \nu$ ;
            end
        end
    end
    if
         $e'/e < \text{TerminationCriterion}$ 
    then
        Converged  $\leftarrow$  true
    end
end
end

```

- ▶ Effective and powerful method for non-linear problems
- ▶ But only if loss is convex
- ▶ Will find local minima and get stuck
- ▶ In those cases, use evolutionary/genetic algorithms

Summary

- ▶ Two different regularisation methods
- ▶ Multivariate regression problems
- ▶ Nonlinear problems (not examinable)
- ▶ The end of regression
- ▶ Next: classification