Machine Learning

Polynomial Regression & Regularisation

Dariush Hosseini

dariush.hosseini@ucl.ac.uk
Department of Computer Science
University College London

Lecture Overview

Learning Outcomes for Today's Lecture

By the end of this lecture you should:

1. Understand that linear regression can lead to **underfitting** which we can remedy by increasing the **complexity** of our representation by using **polynomial regression**
2. Understand that an increase in the complexity of the representation is associated with **overfitting**
3. Know that we can use **regularisation** to overcome overfitting

Lecture Overview

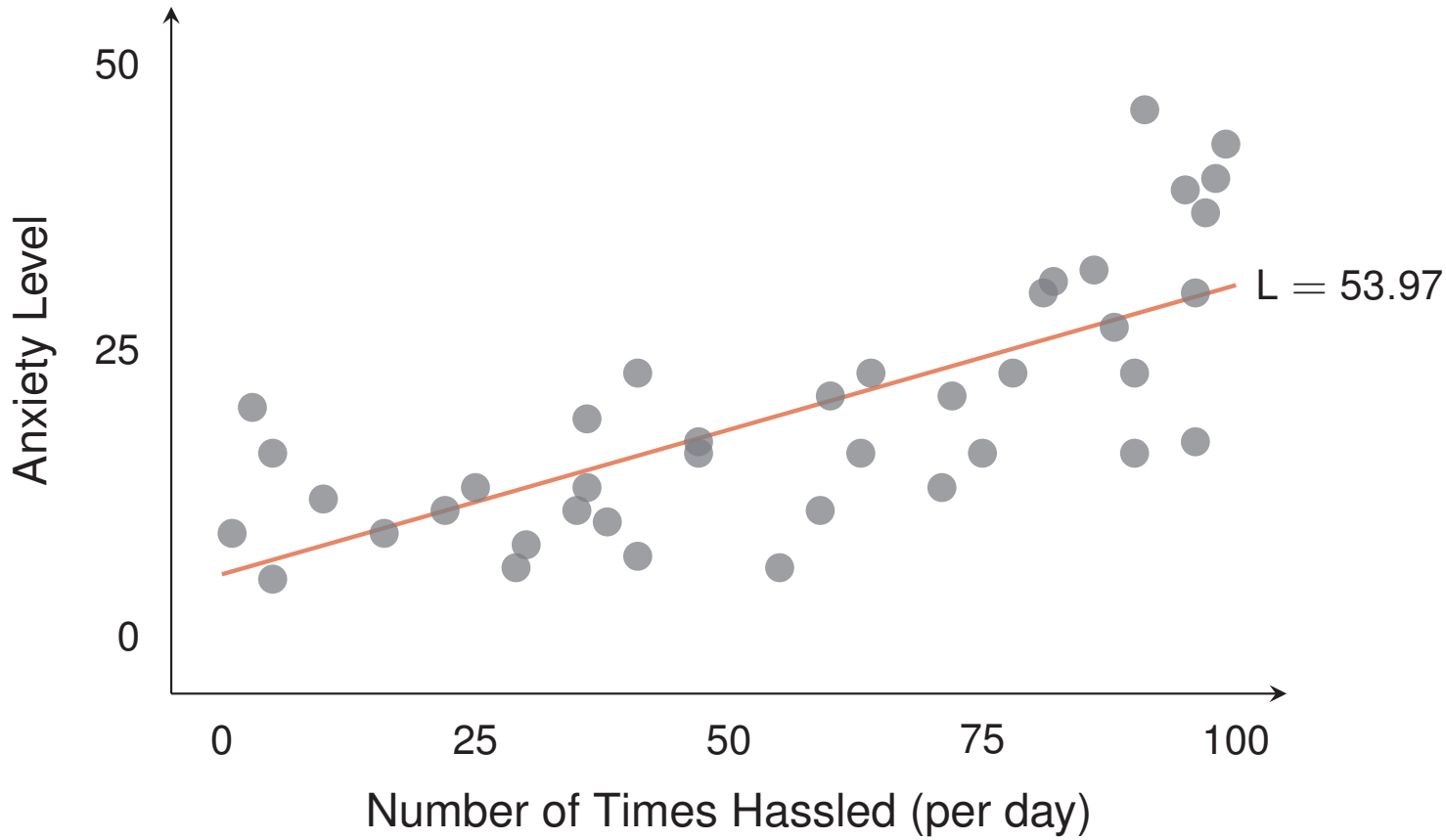
Motivation

- In the last lecture we encountered **linear regression** and in particular **OLS**.
- We restricted ourselves to a **representation** in which the output variable was a **linear function** of the input variables
- But not all relationships are linear in this way...

Underfitting

- ...If we restrict ourselves to a representation which is not rich enough to encompass the true input/output relationship, we will **underfit** the data
- An underfitted model is a model where some parameters or terms that would appear in a correctly specified model are missing
- To remedy this we need to extend our representation somehow to accommodate more functional **complexity**
- **Polynomial Regression** provides one mechanism for doing this

Example


50
25
0
Anxiety Level
0
25
50
75
100
Number of Times Hassled (per day)
L = 53.97

Lecture Overview

Polynomials in One Attribute Variable

- Any polynomial function of one variable can be written in the form:

$$w_0 + w_1 x + w_2 x^2 + \ldots + w_k x^k$$

  where $w_0, \ldots, w_k$ are constants and $k \in \mathbb{N}$ is the **degree** of the polynomial

- Since the model is **linear** in terms of the unknown parameters, we can use the mechanics of linear regression to find the values of $w_0, \ldots, w_k$ and so produce a polynomial fit to our data

Notation

- **Training Sample**:

$$\{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$$

- **Representation**:

$$\mathcal{F} = \left\{ f(x) = \sum_{j=0}^{k} w_j x^j \,\middle|\, k \in \mathbb{N}, w_j \in \mathbb{R} \right\}$$

- **Matrix Notation**:

$$\mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ . \\ . \\ y^{(n)} \end{bmatrix} \mathbf{X} = \begin{bmatrix} 1 & x^{(1)} & (x^{(1)})^2 & . & . & (x^{(1)})^k \\ 1 & x^{(2)} & (x^{(2)})^2 & . & . & (x^{(2)})^k \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 1 & x^{(n)} & (x^{(n)})^2 & . & . & (x^{(n)})^k \end{bmatrix} \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ . \\ . \\ w_k \end{bmatrix}$$

OLS Solution

- Only change from last lecture is that:

$$(x^{(i)})^j \longleftarrow (x_j^{(i)})$$

- So we can wheel out our OLS machinery to generate a solution via the normal equations:

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

OLS Solution

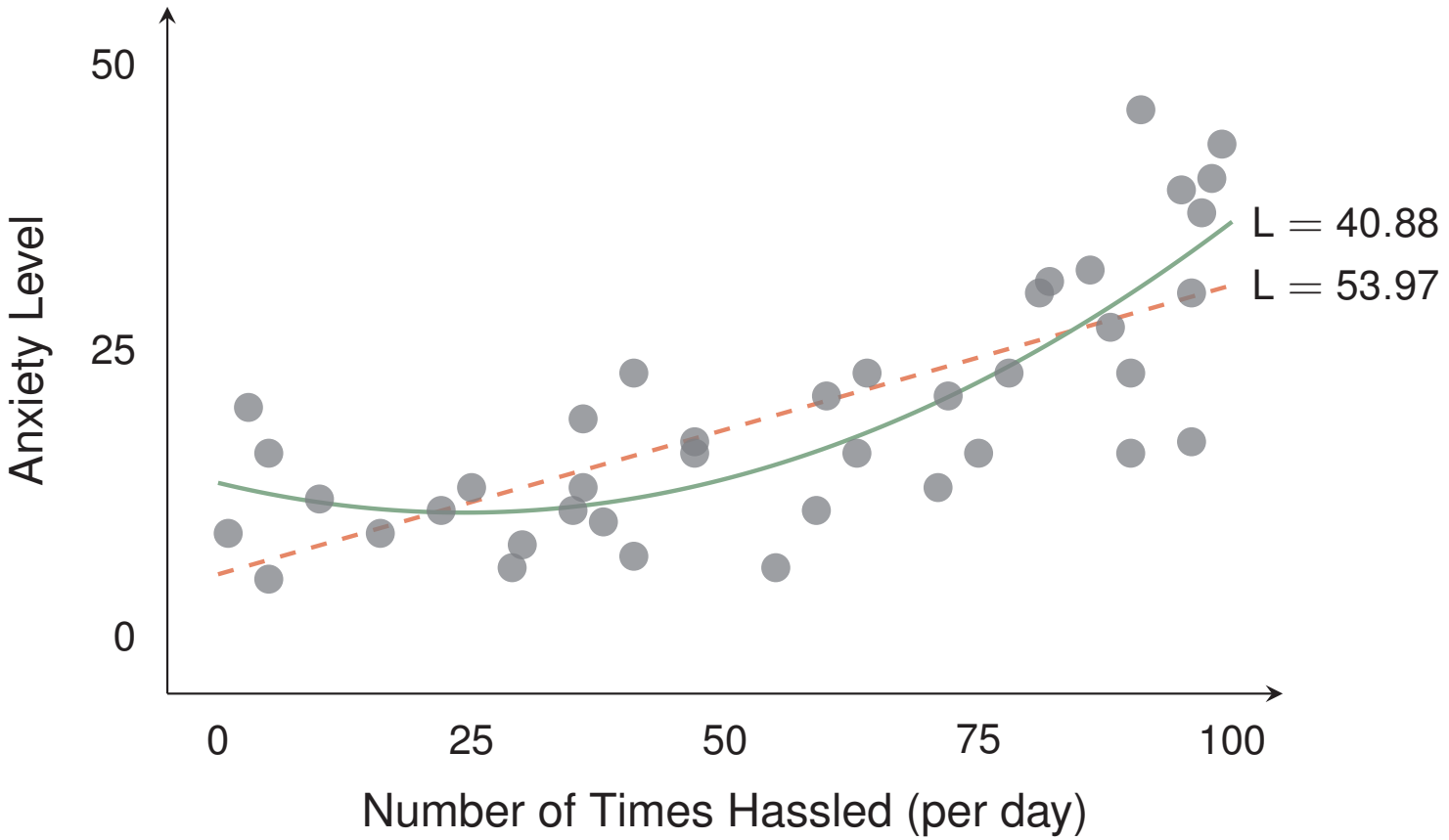
- Of course we need to ensure that we adapt our test data in a similar way
- For a new instance, z, our test attributes will be:

$$\widehat{\mathbf{z}} = [1, z, z^2, ..., z^k]^T$$

- And our output prediction will be:

$$\mathbf{w} \cdot \widehat{\mathbf{z}}$$

Example


50
25
0
Anxiety Level
0
25
50
75
100
Number of Times Hassled (per day)
L = 40.88
L = 53.97

Multiple Attributes

- Size of the design matrix, $\mathbf{X}$, grows as the dimensionality of the input data, and the degree of the polynomial, increase.
- For example, in 2-dimensions (with input data, $\mathbf{x} = [1, x_1, x_2]^T$), a complete polynomial of degree k would be given by:

$$f(\mathbf{x}) = \sum_{l=0}^{k} \sum_{i=0}^{l} \sum_{j=0}^{l-i} w_l {x_1}^i {x_2}^j \qquad \text{where: } (i+j) \leqslant l$$

- The total number of terms (or **multinomial coefficients**) in this polynomial is $(k+1)(k+2)/2$.

Feature Maps

- What we have really done here is to model a set of non-linear functions using a linear technique (**OLS**)
- We do this by performing a polynomial mapping of our input
- But we can do this more generally for many sorts of functional mapping...

Feature Maps

- In general we can characterise our representation by a **feature map**, $\phi : \mathbf{x} \mapsto \phi(\mathbf{x})$ to give:

$$\mathcal{F} = \{f(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x})\}$$

- ϕ can become very complex, even infinite-dimensional

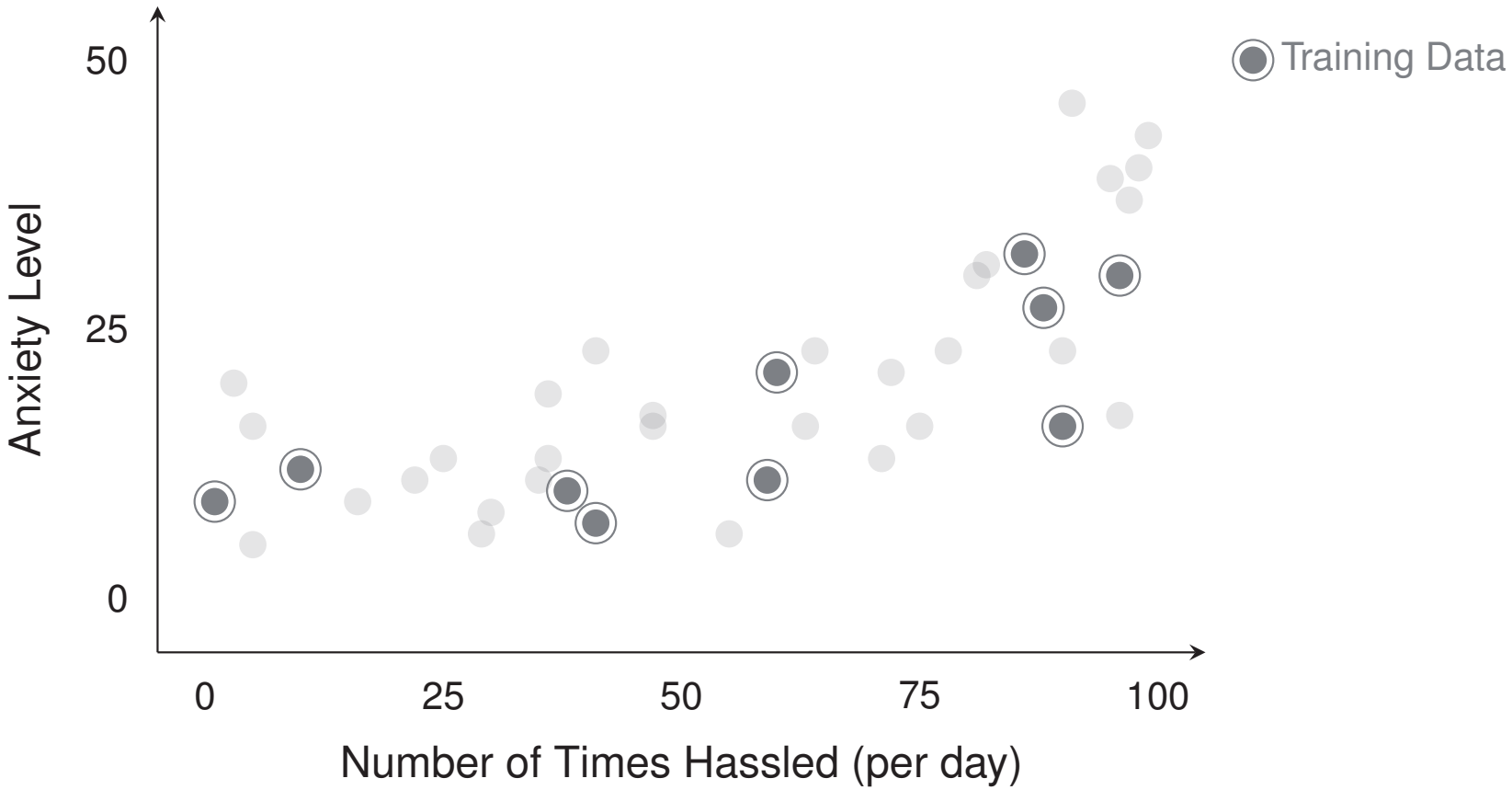
- But if we are careful, we can handle this complexity to make predictions without ever having to:
  - explicitly perform the ϕ-mapping
  - evaluate the weight vector, $\mathbf{w}$, directly

Lecture Overview

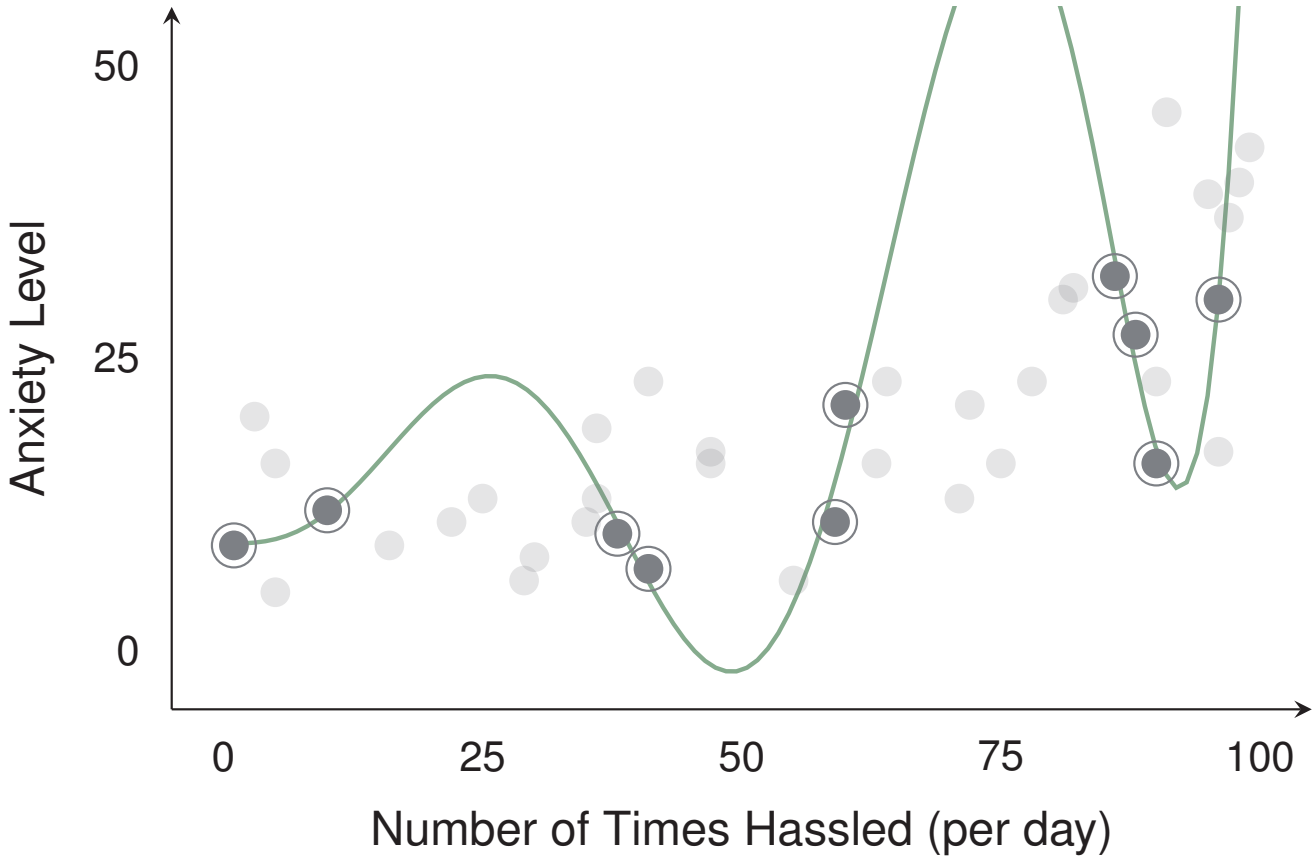
Overfitting

- The benefit of increasing functional complexity is that we can remedy underfitting bias
- The drawback is that we now have the scope to **overfit** the data
- Overfitting occurs when a model fits the residual variation (i.e. the noise) as if that variation represented underlying model structure
  - An overfitted model contains more parameters than can be justified by the data
  - It will achieve good **in-sample** (training) performance, but will generalise poorly **out-of-sample**
- Let's return to our example, but now we are only given a small subset of 10 instances of the data to train our hypothesis on:

Reduced Training Set



Reduced Training Set



Polynomial Overfitting

- Clearly with a high enough degree polynomial we can always reduce our training error to zero
- ...But the function which we learn is likely to perform badly on new data points
- We have started to fit the **noise** rather than the **signal**

Empirical v Generalisation Error

- The problem comes from focusing too much on the **training** data, $\mathcal{S} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$, (with elements drawn from a **data-generating distribution** $\mathcal{D}$)...
- ...So far we have only considered **empirical** evaluation functions
- In machine learning we are really interested in how well our selected function will perform **out-of-sample**
- But because in general we can never see all out-of-sample data, then we don't have access to the evaluation function that we need

Empirical v Generalisation Error

- For a given loss measure, $\mathcal{E}$, applied to a given model, f:
  - We are really interested in the **generalisation error**, $\mathbb{E}_{\mathcal{D}}[\mathcal{E}(f(\mathcal{X}), \mathcal{Y})]$
  - ...But we can only observe the **empirical** (or **training**) **error**, $\mathbb{E}_{\mathcal{S}}[\mathcal{E}(f(\mathcal{X}), \mathcal{Y})] = \frac{1}{n}\sum_{i=1}^{n} \mathcal{E}(f(\mathbf{x}^{(i)}), y^{(i)})$
  - Learning by minimising this empirical error alone is know as **Empirical Risk Minimisation** (**ERM**)

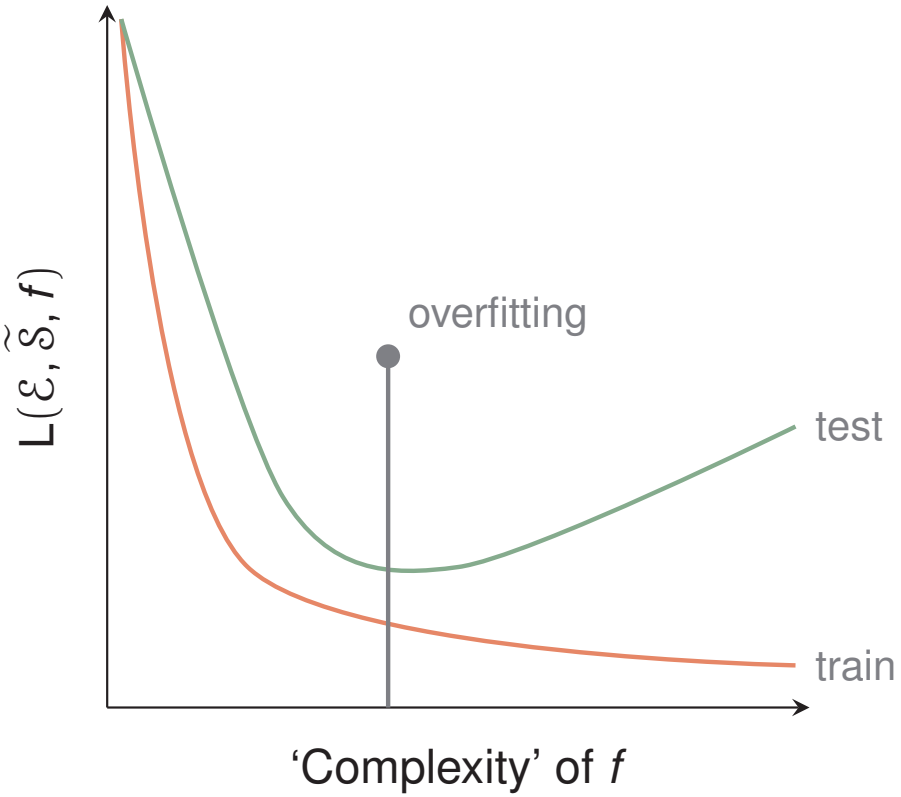
- Much of machine learning is devoted to creating loss functions which approximate the generalisation error in a way that is:
  - Plausibly **close to it**
  - **Tractable** to optimise

Learning Curves

- We can get a sense of how well we have achieved this by evaluating how well our prediction function performs on out-of-sample **test data**, and comparing this with its performance on in-sample **training data**:

Learning Curves

- For a given loss measure, $\mathcal{E}$, we typically see this sort of loss function, L, when applied to a (training or test) data set, $\widetilde{\mathcal{S}}$, and a particular model, f:



Overfitting & Convexity

- Recall that the OLS analytical solution is given by the **normal equations**:

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

- But, for $\mathbf{X} \in \mathbb{R}^{n\times(k+1)}$, if $k+1 > n$ then $\mathbf{X}^T\mathbf{X}$ is **non-invertible**
- We can also show that $\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$ is not **strictly convex** using the definition of strict convexity from the last lecture

Overfitting & Convexity

- These associated facts indicate that we have too few equations to specify the number of unknowns
- In other words we can't find a **unique** OLS solution
- We need a further condition to allow us to:
  - Specify a unique solution
  - Manage complexity
  - Prevent overfitting

Lecture Overview

Regularisation

- **Regularisation** manages model complexity and hence potential for overfitting
- Intuitive motivation:
  - Larger magnitude weights are somehow related to higher complexity
  - Regularisation remedies this by **shrinking** some of the weights towards zero
  - It does this by imposing constraints on the values of **w**
  - Or, equivalently, by restricting the representation

Regularisation

- We will investigate two regularisation techniques, applied to extend OLS linear regression:
  - ℓ_2 regularisation and **Ridge Regression**
  - ℓ_1 regularisation and the **LASSO**

Ridge Regression

- In ridge regression we optimise the usual empirical squared error loss function
  - subject to a constraint on the ℓ_2-norm of the weight vector, $\|\mathbf{w}\|_2^2 = \sum_{j=0}^{k} w_j^2$

- Thus our optimisation problem becomes:

$$\begin{aligned} \underset{\mathbf{w}}{\text{argmin}} \quad & \frac{1}{2} \sum_{i=1}^{n} \left(y^{(i)} - \mathbf{w} \cdot \mathbf{x}^{(i)} \right)^2 \\ \text{subject to:} \quad & \|\mathbf{w}\|_2^2 \leqslant t \end{aligned}$$

For some threshold t

Ridge Regression

- This is equivalent to performing our optimisation with respect to the empirical squared error loss...
- ...but restricting the $\mathbf{w}$ over which we may search for a solution by restricting our representation such that:

$$\mathcal{F} = \left\{ f : \mathbf{x} \mapsto \mathbf{w} \cdot \mathbf{x} \middle| \|\mathbf{w}\|_2^2 \leqslant t, \mathbf{w} \in \mathbb{R}^{k+1} \right\}$$

Ridge Regression

- Using the theory of **Lagrange Multipliers** (see Appendix) we may re-write our optimisation problem as the following problem:

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \lambda) = \mathbf{0}$$

$$\begin{aligned} \text{subject to:} \quad & \|\mathbf{w}\|_2^2 - t \leqslant 0 \\ & \lambda \geqslant 0 \\ & \lambda(\|\mathbf{w}\|_2^2 - t) = 0 \end{aligned}$$

- Where we define the **Lagrangian** function as:

$$\mathcal{L}(\mathbf{w}, \lambda) = \frac{1}{2} \sum_{i=1}^{n} \left(y^{(i)} - \mathbf{w} \cdot \mathbf{x}^{(i)} \right)^2 + \frac{\lambda}{2} \left(\|\mathbf{w}\|_2^2 - t \right)$$

- In general a constrained optimisation is not straightforward to solve

Ridge Regression

- But here, because we have only one Lagrange multiplier to consider, matters are not so difficult:
- Either: $\lambda = 0$
  - Then: $\|\mathbf{w}\|_2^2 < t$
  - So we have set t so high that the constraint is redundant, and the solution is simply the usual non-regularised one
  - We must set t lower in order for regularisation to become effective
- Or: $\lambda > 0$
  - Then: $\|\mathbf{w}\|_2^2 = t$
  - In this case we have an equality constraint and we can proceed by optimising the unconstrained Lagrangian (see Appendix)

Ridge Regression

- In this case the solution gives a **correspondence** between λ and t
- We can solve the problem for a range of λ values and this is equivalent to solving for a range of t values
- For any given t, if we look for **stationarity** w.r.t. λ, (i.e. $\nabla_\lambda \mathcal{L}(\mathbf{w}, \lambda) = 0$), we end up with a corresponding, optimal, $\lambda = \lambda^*$
- Also note that $\mathcal{L}(\mathbf{w}, \lambda)$ is convex wrt $\mathbf{w}$ so the stationarity condition ($\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \lambda) = \mathbf{0}$) will be achieved at the minimal point of $\mathcal{L}(\mathbf{w}, \lambda^*)$ with respect to $\mathbf{w}$

Ridge Regression

- Substituting this back into the Lagrangian leaves us with the following problem to solve:

$$\begin{aligned} \underset{\mathbf{w}}{\text{argmin}} \quad & \frac{1}{2}\sum_{i=1}^{n}\left(y^{(i)} - \mathbf{w}\cdot\mathbf{x}^{(i)}\right)^2 + \frac{\lambda^*}{2}\left(\|\mathbf{w}\|_2^2 - t\right) \\ \text{subject to:} \quad & \lambda^* \geqslant 0 \end{aligned}$$

- Which is, of course, equivalent to:

$$\begin{aligned} \underset{\mathbf{w}}{\text{argmin}} \quad & \frac{1}{2}\sum_{i=1}^{n}\left(y^{(i)} - \mathbf{w}\cdot\mathbf{x}^{(i)}\right)^2 + \frac{\lambda^*}{2}\|\mathbf{w}\|_2^2 \\ \text{subject to:} \quad & \lambda^* \geqslant 0 \end{aligned}$$

Ridge Regression

- This objective becomes our new evaluation function, $\widetilde{\mathsf{L}}(\mathcal{E}, \mathcal{S}, f_{\mathbf{w}})$
- We can write this in full matrix form:

$$\widetilde{\mathsf{L}}(\mathcal{E}, \mathcal{S}, f_{\mathbf{w}}) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$$

- And then take the derivative:

$$\nabla_{\mathbf{w}}\widetilde{\mathsf{L}} = \mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathbf{y} + \lambda\mathbf{w}$$

Ridge Regression

- We set this equal to zero to find the optimal solution:

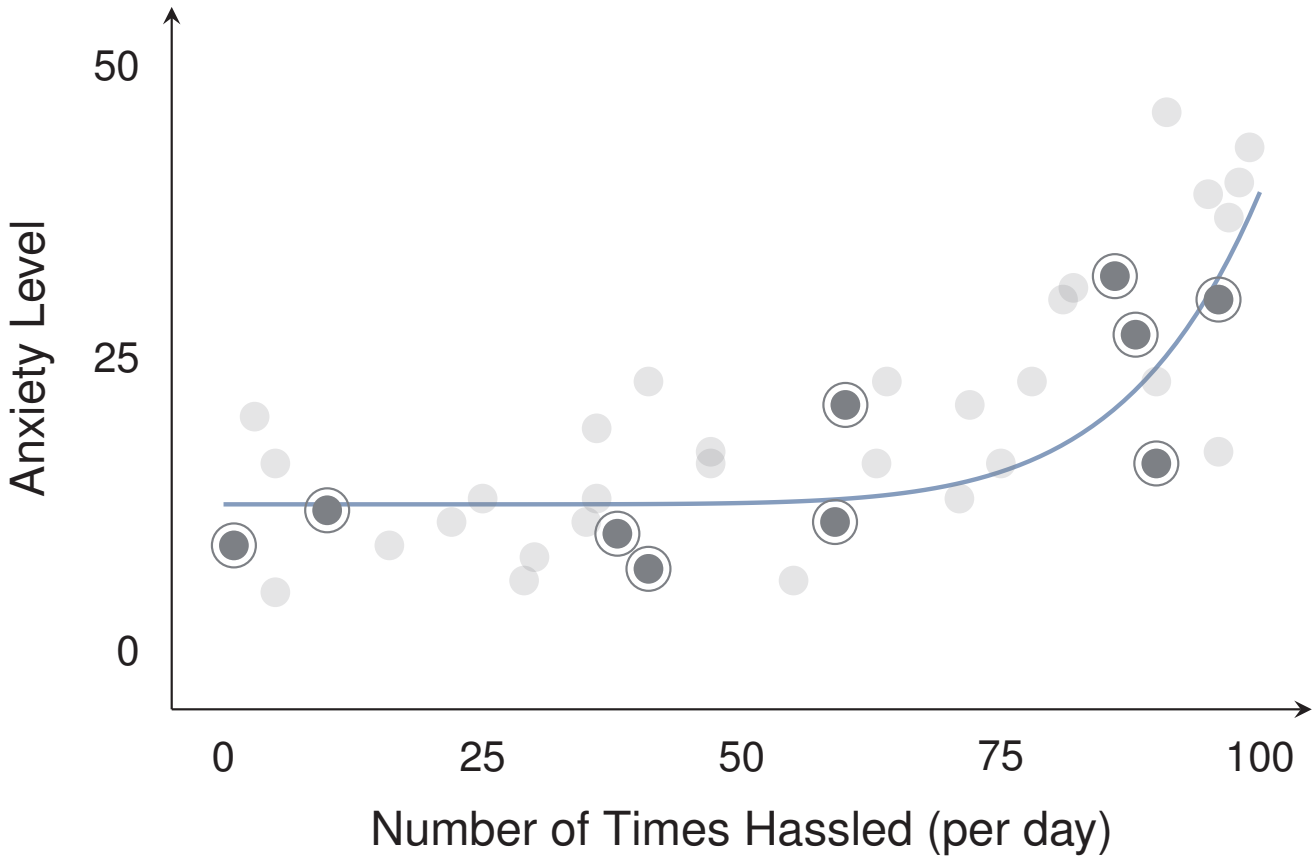
$$\begin{aligned}\mathbf{X}^T\mathbf{X}\mathbf{w}_{RR} - \mathbf{X}^T\mathbf{y} + \lambda\mathbf{w}_{RR} &= 0 \\ (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\mathbf{w}_{RR} &= \mathbf{X}^T\mathbf{y} \\ \mathbf{w}_{RR} &= (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}\end{aligned}$$

- These are the **revised normal equations**

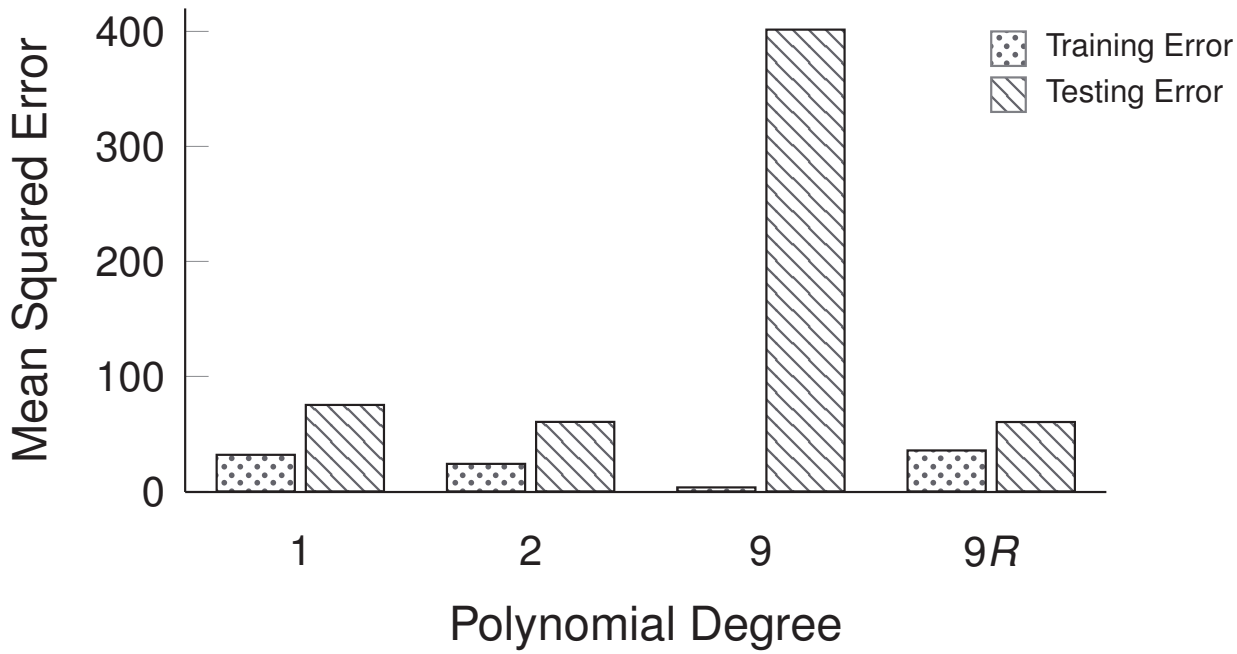
Ridge Regression

- The $\lambda\mathbf{I}$ term forms a **ridge**
- $(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})$ is always **invertible** because its rank is always equal to $k + 1$
- The Hessian, $\mathcal{H} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})$, is therefore **positive definite**
- So $\widetilde{\mathrm{L}}$ is **strictly convex** and gradient descent will converge to a **unique** solution

9-degree Regularised Polynomial



Comparing Training and Test Error



Complexity Tuning

- Together the regularisation parameter, λ, and the polynomial order, k, are complexity tuning **hyperparameters**
  - High λ, low $k \longrightarrow$ **low functional complexity** $\longrightarrow$ **underfitting**
  - Low λ, high $k \longrightarrow$ **high functional complexity** $\longrightarrow$ **overfitting**
- We will return to the question of how we are to set such hyperparameters in a future lecture...

Underfitting - Revisited

- Recall that an underfitted model is a model where some parameters or terms that would appear in a correctly specified model are missing
- We say that an underfitted model depends on a learning algorithm whose representation is too **biased**
- Thus there is a connection between underfitting, low functional complexity and high bias
- High bias learning algorithms often exhibit poor training set performance - in other words they are poor empirical risk minimisers

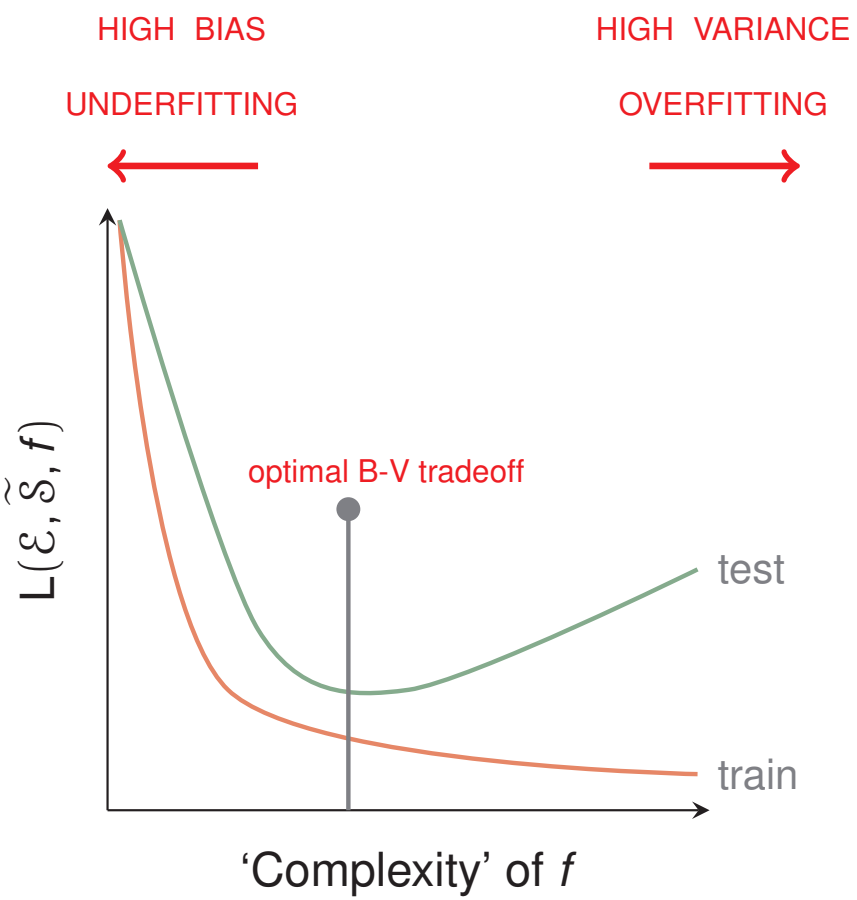
Overfitting - Revisited

- Recall that an overfitted model contains more parameters than can be justified by the data
- We say that an overfitted model depends on a learning algorithm which has too much **variance**
- Thus there is a connection between overfitting, high functional complexity and high variance
- High variance learning algorithms often exhibit excellent training set performance...but poor generalisation performance

The Bias Variance Tradeoff

- There seems to be a **tradeoff** between the bias and variance of a learning algorithm
- We will never be able to completely reduce both effects (in the absence of complete population data)
- Often, as we increase the complexity of the model class within our learning algorithm the bias decreases but our variance increases as the hypothesis space grows
- This helps explains why **regularisation** often works so well - it acts to bias our learning algorithm, but also reduces its variance

Learning Curves



Feature Selection

- In ridge regression, if we have a large number of weights, then in general all of these weights will be included in the final model
- The regularised weights can shrink to almost zero but will rarely actually reach zero
- This can make **model interpretation** difficult when the number of weights is large

Feature Selection

- Often we would like a way of selecting the most important features to make our model more transparent
- In general we can do this by performing a **combinatorial search** over $\{w_j\}_{j=0}^{k}$...
- ...then selecting the subset of w_j which gives the optimal value on our loss function
- But this is not **tractable**...Is there an alternative?

LASSO

- In the LASSO we optimise the usual empirical squared error loss function
  - subject to a constraint on the ℓ_1-norm of the weight vector, $\|\mathbf{w}\|_1 = \sum_{j=0}^{k} |w_j|$
  - as we will see, this has the effect of shrinking some weights more aggressively to zero
- Thus our optimisation problem becomes:

$$\begin{aligned} \underset{\mathbf{w}}{\text{argmin}} \quad & \frac{1}{2} \sum_{i=1}^{n} \left(y^{(i)} - \mathbf{w} \cdot \mathbf{x}^{(i)} \right)^2 \\ \text{subject to:} \quad & \|\mathbf{w}\|_1 \leqslant t \end{aligned}$$

For some threshold t

LASSO

- This is equivalent to performing our optimisation with respect to the empirical squared error loss
  - ...but restricting the $\mathbf{w}$ over which we may search for a solution by restricting our representation such that:

$$\mathcal{F} = \left\{ f : \mathbf{x} \mapsto \mathbf{w} \cdot \mathbf{x} \,\middle|\, \|\mathbf{w}\|_1 \leqslant t, \mathbf{w} \in \mathbb{R}^{k+1} \right\}$$

- Using a similar argument to before[1] we may re-write our optimisation problem as:

$$\underset{\mathbf{w}}{\operatorname{argmin}} \quad \frac{1}{2} \sum_{i=1}^{n} \left(y^{(i)} - \mathbf{w} \cdot \mathbf{x}^{(i)} \right)^2 + \frac{\lambda}{2} \|\mathbf{w}\|_1$$

Where $\lambda \geqslant 0$ is a Lagrange multiplier

[1] Note that here the objective is not **differentiable**, so we must use **subgradients** when reformulating the problem

LASSO

- This becomes our new evaluation function, $\widetilde{\mathrm{L}}(\mathcal{E}, \mathcal{S}, f_{\mathbf{w}})$, which we wish to optimise
- But this function is not **differentiable**, so we can't look for stationary points in order to solve, as we did for RR
- We need a numerical approach, for example:
  - Proximal Gradient Methods

Why does LASSO lead to feature selection?

- We are seeking to optimise the following function:

$$\widetilde{\mathsf{L}}(\mathcal{E}, \mathcal{S}, f_{\mathbf{w}}) = \mathsf{L}(\mathcal{E}, \mathcal{S}, f_{\mathbf{w}}) + \frac{\lambda}{2}\|\mathbf{w}\|_1$$

  Where: $\mathsf{L}(\mathcal{E}, \mathcal{S}, f_{\mathbf{w}}) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$

- Let's investigate its behaviour around $w_j = 0$:
  - We will investigate the **subgradients**:

- For $w_j > 0$, the subgradient is: $\frac{\partial \mathsf{L}}{\partial w_j} + \frac{\lambda}{2}$

- For $w_j < 0$, the subgradient is: $\frac{\partial \mathsf{L}}{\partial w_j} - \frac{\lambda}{2}$

Why does LASSO lead to feature selection?

- So at $w_j = 0$ we are at an optimal point if:

$$\frac{\partial L}{\partial w_j} + \frac{\lambda}{2} > 0 \qquad \text{and;} \qquad \frac{\partial L}{\partial w_j} - \frac{\lambda}{2} < 0$$

$$\implies \qquad \frac{\partial L}{\partial w_j} > -\frac{\lambda}{2} \qquad \text{and;} \qquad \frac{\partial L}{\partial w_j} < \frac{\lambda}{2}$$

- This happens if:

$$-\frac{\lambda}{2} < \frac{\partial L}{\partial w_j} < \frac{\lambda}{2}$$

- As λ increases, the range of values of $\frac{\partial L}{\partial w_j}$ for which w_j remains at 0 if it is already there, gets larger...

Why does LASSO lead to feature selection?

- Compare this with ridge regression, where we are at an optimal point if:

$$\frac{\partial L}{\partial w_j} + \lambda w_j = 0$$

- At $w_j = 0$ this happens if:

$$\frac{\partial L}{\partial w_j} = 0$$

- Clearly this represents a more restrictive range of values of $\frac{\partial L}{\partial w_j}$ for which optimality holds at a point for which $w_j = 0$ than for the LASSO...

LASSO: Problems

- When we are dealing with high dimensional data with few examples ($n < (k + 1)$) it turns out that we can prove the LASSO selects at most n variables before it saturates
- If there is a group of highly correlated variables then the LASSO tends to select one from the group and ignore the others
- The LASSO optimisation problem is not necessarily strictly convex so, in general, it will not yield a unique solution

Elastic Net

- **Elastic Net Regularisation** seeks to remedy these shortcomings, by combining Ridge Regression and the LASSO

- The optimisation problem associated with the Elastic Net can be written as:

$$\underset{\mathbf{w}}{\text{argmin}} \quad \frac{1}{2}\sum_{i=1}^{n}\left(y^{(i)} - \mathbf{w}\cdot\mathbf{x}^{(i)}\right)^2 + \frac{\lambda}{2}\left(\alpha\|\mathbf{w}\|_2^2 + (1-\alpha)\|\mathbf{w}\|_1\right)$$

Where: $0 \leqslant \alpha \leqslant 1, \lambda \geqslant 0$

Elastic Net

- The Elastic Net has the following beneficial properties:
  - Its objective is strictly convex, and so optimal Elastic Net weights are **unique**
  - The presence of the ℓ_1-norm term in the objective function promotes **sparsity**...
  - ...However, the presence of the ℓ_2-norm term encourages a **grouping effect**...
  - ...And removes the limitation on the number of selected variables

Recap

	Representation	Loss Measure	Optimisation	Solution
OLS	$\mathcal{F} = \{\mathbf{w} \cdot \mathbf{x}\}$	$\frac{1}{2}\sum_{i=1}^{n}(y^{(i)} - \mathbf{w} \cdot \mathbf{x}^{(i)})^2$	$\underset{\mathbf{w}}{\text{argmin}} \quad \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$	NE & GD
Ridge Regression	$\mathcal{F} = \{\mathbf{w} \cdot \mathbf{x} \mid \|\mathbf{w}\|_2^2 \leqslant t\}$	$\frac{1}{2}\sum_{i=1}^{n}(y^{(i)} - \mathbf{w} \cdot \mathbf{x}^{(i)})^2$	$\underset{\mathbf{w}}{\text{argmin}} \quad \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2$	Revised NE & GD
LASSO	$\mathcal{F} = \{\mathbf{w} \cdot \mathbf{x} \mid \|\mathbf{w}\|_1 \leqslant t\}$	$\frac{1}{2}\sum_{i=1}^{n}(y^{(i)} - \mathbf{w} \cdot \mathbf{x}^{(i)})^2$	$\underset{\mathbf{w}}{\text{argmin}} \quad \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_1$	Numerical Optimisations
Elastic Net	$\mathcal{F} = \{\mathbf{w} \cdot \mathbf{x} \mid \|\mathbf{w}\|_2^2 \leqslant t_1, \|\mathbf{w}\|_1 \leqslant t_2\}$	$\frac{1}{2}\sum_{i=1}^{n}(y^{(i)} - \mathbf{w} \cdot \mathbf{x}^{(i)})^2$	$\underset{\mathbf{w}}{\text{argmin}} \quad \frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \frac{\lambda}{2}\left(\alpha\|\mathbf{w}\|_2^2 + (1-\alpha)\|\mathbf{w}\|_1\right)$	Numerical Optimisations

- But why is regularisation a sensible tool for managing complexity?

Motivation

- We can motivate regularisation in a number of ways, e.g.:
  - **Bayesian linear regression**:
    - **w** is modelled as a **random variable**
    - **MAP** solution to weight optimisation will result in a **regularised objective**
    - Depending on which **distributional form** we select for **w**, we can derive ridge regression or the LASSO
  - **PAC approach**:
    - Model a worst case **probabilistic bound** on the generalised evaluation function, $\mathbb{E}_{\mathcal{D}}[\mathrm{L}]$, given some function class
    - Bound will take a similar form to: $\mathbb{E}_{\mathcal{S}}[\mathrm{L}] +$ Regularisation Term, and its optimisation will result in a **regularised objective**
    - Depending on which **function class** we select, we can derive ridge regression or the LASSO

Lecture Overview

Summary

1. Linear Regression can be extended to non-linear function classes to overcome **underfitting** using **feature maps** such as the **polynomial basis**.

2. If we do not constrain the **complexity** of the classes of such feature maps we can **overfit** our data and risk encountering problems for which we have no **unique** solution.

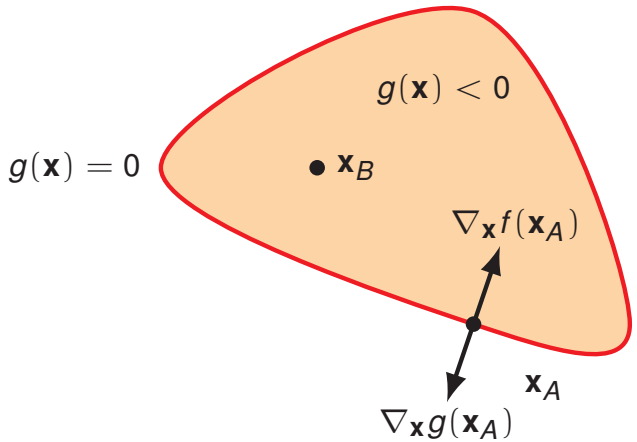
3. **Regularisation** is a way of controlling this overfitting. In particular **Ridge Regression** and the **LASSO** are ways of implementing regularisation in a **convex** fashion.

In the next lecture we will move on from regression and begin to look at another supervised learning task: **classification**

Lecture Overview

Notation[2]

- $\mathbf{x} \in \mathbb{R}^n$
- $f : \mathbb{R}^n \to \mathbb{R}$ is the function over which we wish to optimise $\mathbf{x}$
- $g(\mathbf{x}) = 0$ represents an $(n-1)$ dimensional surface constraint
- $n = 2$ dimensional illustration (with $g(\mathbf{x}_A) = 0$, and $g(\mathbf{x}_B) < 0$):



[2] Content and illustrations based on Bishop, 'Pattern Recognition & Machine Learning' [2008]

Equality Constraints: Problem

■

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & f(\mathbf{x}) \\ \text{subject to:} \quad & g(\mathbf{x}) = 0 \end{aligned}$$

Note that the functions f and g can be convex or nonconvex in general.

Equality Constraints: Observations

- $\nabla_{\mathbf{x}} g(\mathbf{x})$ is orthogonal to the surface defined by $g(\mathbf{x})$:
  - Because, if we denote any direction along the surface $g(\mathbf{x})$ by $\widehat{\mathbf{u}}$, then because the directional derivative along the direction of the surface must be zero 0, $\nabla_{\mathbf{x}} g(\mathbf{x}) \cdot \widehat{\mathbf{u}} = 0$.
- The optimal point, $\mathbf{x}^*$ must have the property that $\nabla_{\mathbf{x}} f(\mathbf{x}^*)$ is orthogonal to the constraint surface:
  - Because, otherwise $f(\mathbf{x})$ could decrease for movements along the surface.

Equality Constraints: Lagrange Multiplier

- Thus $\nabla_{\mathbf{x}} f(\mathbf{x})$ and $\nabla_{\mathbf{x}} g(\mathbf{x})$ must be parallel, i.e.:

$$\nabla_{\mathbf{x}} f(\mathbf{x}) + \lambda \nabla_{\mathbf{x}} g(\mathbf{x}) = 0 \quad \text{for some:} \quad \lambda \neq 0$$

Here λ is a so-called **Lagrange multiplier**

Equality Constraints: Lagrangian

- Let us define the **Lagrangian** function, $\mathcal{L}$, as follows:

$$\mathcal{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x})$$

- Then:

$$\nabla_{\mathbf{x}} \mathcal{L} = \nabla_{\mathbf{x}} f(\mathbf{x}) + \lambda \nabla_{\mathbf{x}} g(\mathbf{x})$$
$$\nabla_{\lambda} \mathcal{L} = g(\mathbf{x})$$

Equality Constraints: Problem reformulation

- Seek stationary solutions $(\mathbf{x}^*, \lambda^*)$ which satisfy the following:

$$\nabla_{\mathbf{x}} \mathcal{L} = \mathbf{0}$$
$$\nabla_{\lambda} \mathcal{L} = 0$$

- Thus we have transformed our problem into an **unconstrained** optimisation problem

- ...But note that the optimal solution will be a **saddle point**. Why?

Equality Constraints: Saddle Point

- Consider the Hessian matrix for the Lagrangian:

$$\mathcal{H}(\mathbf{x}, \lambda) = \begin{bmatrix} \nabla^2_{\mathbf{x}} f(\mathbf{x}) + \lambda \nabla^2_{\mathbf{x}} g(\mathbf{x}) & \nabla_{\mathbf{x}} g(\mathbf{x}) \\ \nabla_{\mathbf{x}} g(\mathbf{x})^T & 0 \end{bmatrix}$$

- Now consider the quadratic form $\boldsymbol{\alpha}^T \mathcal{H}(\mathbf{x}, \lambda) \boldsymbol{\alpha}$, where $\boldsymbol{\alpha} = [\mathbf{a}, b]^T$, for all $\mathbf{a} \in \mathbb{R}^n$ and $b \in \mathbb{R}$:

$$\boldsymbol{\alpha}^T \mathcal{H}(\mathbf{x}, \lambda) \boldsymbol{\alpha} = \mathbf{a}^T \left(\nabla^2_{\mathbf{x}} f(\mathbf{x}) + \lambda g(\mathbf{x})^2 g(\mathbf{x}) \right) \mathbf{a} + 2b\mathbf{a} \cdot \nabla_{\mathbf{x}} g(\mathbf{x})$$

- Clearly if $\nabla_{\mathbf{x}} g(\mathbf{x})$ is finite then it is always possible to select $\mathbf{a}$, b such that the second term dominates the first in magnitude and can be made either positive or negative.

Equality Constraints: Saddle Point

- Thus $\mathcal{H}(\mathbf{x}, \lambda)$ is **indefinite** and the stationary points for $\mathcal{L}(\mathbf{x}, \lambda)$ are thus saddle points
- Note that this makes the use of **gradient descent** as an optimisation procedure problematic.
  But alternatives exist (e.g. **Newton's Method**)

Inequality Constraints: Problem

- $$\begin{aligned} \min_{\mathbf{x} \in \mathbb{R}^n} \quad & f(\mathbf{x}) \\ \text{subject to:} \quad & g(\mathbf{x}) \leqslant 0 \end{aligned}$$

- Two types of solution are possible:

Inequality Constraints: Inactive Constraint

- $\mathbf{x}^*$ lies in $g(\mathbf{x}) < 0$
- Stationary condition $\nabla_{\mathbf{x}} f(\mathbf{x}) = \mathbf{0}$
- Which is equivalent to:

$$\nabla_{\mathbf{x}} \mathcal{L} = \mathbf{0} \quad \text{with:} \quad \lambda = 0 \tag{2}$$

Inequality Constraints: Active Constraint

- $\mathbf{x}^*$ lies on $g(\mathbf{x}) = 0$
- Since the solution does not lie in $g(\mathbf{x}) < 0$ then $f(\mathbf{x})$ will only be minimal if $\nabla_{\mathbf{x}} f(\mathbf{x})$ points towards the $g(\mathbf{x}) < 0$ region. Thus:

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = -\lambda \nabla_{\mathbf{x}} g(\mathbf{x}) \quad \text{for} \quad \lambda > 0$$

- Which is equivalent to:

$$\nabla_{\mathbf{x}} \mathcal{L} = \mathbf{0} \quad \text{with:} \quad \lambda > 0 \tag{3}$$

Inequality Constraints: Problem Reformulation

- Using equations (2) & (3), we can solve our problem by seeking stationary solutions $(\mathbf{x}^*, \lambda^*)$ which satisfy the following:

$$\nabla_{\mathbf{x}} \mathcal{L} = \mathbf{0}$$

$$\text{subject to:} \quad \begin{cases} g(\mathbf{x}) \leqslant 0 \\ \lambda \geqslant 0 \\ \lambda g(\mathbf{x}) = 0 \end{cases}$$

- These conditions are known as the **Karush Kuhn Tucker** (KKT) conditions.

Inequality Constraints: Complementary Slackness

- $\lambda g(\mathbf{x}) = 0$ is satisfied for both the **active** and **inactive cases**, and is known as the **complementary slackness** condition

- It is equivalent to:

$$\begin{aligned} \lambda > 0 \quad &\Longrightarrow \quad g(\mathbf{x}) = 0 \\ g(\mathbf{x}) < 0 \quad &\Longrightarrow \quad \lambda = 0 \end{aligned}$$