What is Machine Learning?

Oxford Spring School in Advanced Research Methods, 2021

Dr Thomas Robinson, Durham University

Day 1/5

Introduction

This course

5×3 hour sessions, covering:

1. Introduction to ML and maximum likelihood estimation
2. ML extensions to regression
3. Tree-based methods
4. Neural networks
5. Ensemble methods

The logic:

- Understand the underlying mechanics of parameter estimation
- Starting from a modeling strategy we are likely familiar with...
- ... moving on to those that are algorithmically more complicated
- Building on the same foundational concepts across each day

Balancing a course on machine learning

- ML is a broad and contested domain
  - Within each sub-domain, there is a lot of potential (mathematical) detail
  - Easy to get stuck in the thickets of estimation
- Mindful of how ML applies to practical research problems
- In balancing these objectives, there are lots of ML topics we will not cover:
  - Unsupervised methods
  - Clustering algorithms
  - Text-specific ML models

This course prioritises:

- Intuitive understanding of popular ML methods
- Transferability of fundamentals across ML strategies
- Relevance to our social science research streams

Session structure

Each session will be a mixture of:

- Lecture content and discussion
  - I will leave plenty of opportunities for questions
  - Building in some time for us to think through some applied problems
- Coding walkthroughs (approx. 1 hour)
  - Conducted on RStudio Cloud
  - Hands on experience using different algorithms
  - Many of these are now very easy to implement

Slides and code are available at:
https://github.com/tsrobinson/ox_ML

Today's session

Goals are threefold:

1. Introduce the topic of machine learning
   - What is ML?
   - How do we distinguish it from statistics?
   - What sort of problems might we apply it to?
2. Introduce key conceptual distinctions we will make throughout the course
3. Introduce maximum likelihood estimation
   - A key way in which ML parameters are estimated
   - Build our own logistic regression estimator

What is machine learning?

(Machine) learning and statistics

ML is a vague and contested field:

> *"Machine learning is a subfield of* ***computer science*** *that is concerned with building algorithms which, to be useful, rely on a collection of examples of some phenomenon. . . the process of solving a practical problem by 1) gathering a dataset, and 2)* ***algorithmically*** *building a statistical model based on that dataset." – Burkov 2019*

> *"Statistical learning refers to a set of tools for modeling and understanding* ***complex*** *datasets. It is a recently developed area in statistics and blends with parallel developments in* ***computer science*** *and, in particular, machine learning" – James et al. 2013*

Prediction and inference

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{1i} + e_i$$

ML typically focuses on prediction problems instead of inference problems:

- Inference is concerned with estimating the size/direction of the relationship between variables ($\hat{\boldsymbol{\beta}}$ problems)
- Prediction is concerned with estimating some outcome, using the relationships between variables ($\hat{\boldsymbol{y}}$ problems)

These two facets are clearly connected:

- If we know the size/direction of the relationships, we can predict the outcome
- But we rarely know (or even pretend to know) the true model
- Sometimes we can get good at $\hat{\boldsymbol{y}}$ problems without knowing $\hat{\beta}$

Classification and prediction

We can distinguish two types of prediction problem

- **Prediction** – estimating the value of a continuous variable (sometimes referred to as "regression" problems)
  - E.g. The ideology of a politician in 2D space
  - The number of votes received by a candidate
  - Some interpersonal characteristic of an individual
- **Classification** – estimating which *class* of a category an observation belongs to
  - Party identity (Republican/Democrat/Libertarian/Independent)
  - The topic of a tweet (foreign/domestic, pro/con)

$\hat{Y}$ and $\hat{X}$ problems

We can also think about where the prediction problem lies:

- $\hat{\boldsymbol{y}}$ problems are about the dependent variable
  - To predict an election winner...
  - ... or the probability of revolution...
  - ... or the weather tomorrow
  - These are not necessarily inferential problems
- $\hat{\boldsymbol{X}}$ problems are about independent variables
  - Dimensions of interest that may be important to our theory...
  - ... but which are not directly observable (i.e. latent)
  - We want to make predictions over $\boldsymbol{X}$ so we can test an inferential theory about the relationship between X and y

Supervised vs unsupervised methods

Supervised methods

- Contains labeled examples of observations with corresponding outcomes $\boldsymbol{y}$ – the **training data**
- Use these examples to "learn" the relationship between $\boldsymbol{y}$ and $\boldsymbol{X}$
- Then predict $\boldsymbol{y}$ for a *new* **test** dataset $\boldsymbol{X}^{\text{TEST}}$ where $\boldsymbol{y}^{\text{TEST}}$ is not observed

Learning the relationship:

$$\underbrace{\begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}}_{\mathbf{y}^{\text{TRAIN}}} \quad \underbrace{\begin{bmatrix} 3.3 & 1.1 & 0 \\ 2.7 & 0.8 & 0 \\ 1.8 & 0.1 & 1 \\ \vdots & \vdots & \vdots \\ 5 & 1.2 & 0 \end{bmatrix}}_{\mathbf{X}^{\text{TRAIN}}}$$

Predicting on new data

$$\mathbf{X}^{\text{TEST}} = \begin{bmatrix} 3.5 & 1.9 & 1 \\ 5.4 & 0.3 & 0 \\ 1.7 & 0.5 & 1 \end{bmatrix}$$

Machine learning

Expectation: I need a $1m super computer

Reality: It runs in minutes on a personal computer

Figure 1: Google: Tensor Processing Unit server rack

Why machine learning?

Machine learning can be:

- Powerful
- Flexible
- Reduce the burden on the researcher

It helps solve lots of **prediction problems**...

... and can assist in **inference problems** too

- Through $\hat{X}$ and $\hat{Y}$ problems

Machine learning and social science

But ML is not a panacea!

- ML cannot solve problems of poor research design
- And can introduce it's own issues

Twitter apologises for 'racist' image-cropping algorithm

Users highlight examples of feature automatically focusing on white faces over black ones

Minority Report-style tech used to predict crime is 'prejudiced'

In his first interview since becoming surveillance commissioner, Fraser Sampson warns about accuracy of predictive policing technology

Maximum Likelihood Estimation (a gentle introduction)

Notation

Throughout the course, we will follow the notation set out in Burkov (2019):

- θ is a scalar - i.e. a single number
  - E.g., $\theta = 3.141$, $x = 1$ etc.
- $\boldsymbol{\theta}$ is a vector - i.e. an ordered list of scalar values
  - E.g., $\boldsymbol{\theta} = [0.5, 3, 2]$
- $\mathbf{X}$ is a matrix
  - E.g.,

$$\mathbf{X} = \begin{bmatrix} 1 & 5 \\ 24 & -3 \end{bmatrix}$$

  - $\mathbf{x}^{(k)}$ is the kth column of $\boldsymbol{X}$
  - $\mathbf{x_i}$ is the ith row in matrix $\mathbf{X}$
  - $x_i^{(k)}$ is the kth element of the row vector $\boldsymbol{x_i}$

Probability notation

Let p denote a probability distribution *function* that returns the probability of an event or observation:

- $p(A) = 0.5$ means the probability of event A is 0.5
- $0 \leq p(A) \leq 1$
- Conditional probability $p(A|c) = 0.25$ means the probability of A **given** (or conditional on) c is 0.25.
- $p(A \text{ and } B) = p(A)p(B|A)$
- If $p(A) \perp p(B)$, then $p(B|A) = p(B)$
  - So, if $p(A) \perp p(B)$, then $p(A \text{ and } B) = p(A)p(B)$

Notation quiz

What are the following:

1. $\mathbf{a}$
2. y_i
3. $\boldsymbol{\beta}$
4. β
5. $\boldsymbol{\Theta}$

If $p(A) = 0.5$, $p(B) = 0.1$, and $p(B|A) = 0.3$:

6. Is $p(A) \perp p(B)$?
7. What is $p(A \text{ and } B)$?

Bayes Theorem (from a frequentist perspective)

$$\underbrace{P(A|B)}_{\text{Posterior}} = \frac{\overbrace{P(B|A)}^{\text{Likelihood}} \times \overbrace{P(A)}^{\text{Prior}}}{\underbrace{P(B)}_{\text{Evidence}}}$$

We can use Bayes formula to estimate the posterior probability of some parameter $\boldsymbol{\theta}$:

$$p(\boldsymbol{\theta}|\mathbf{X}) \propto p(\mathbf{X}|\boldsymbol{\theta}) \times p(\boldsymbol{\theta}),$$

where $\mathbf{X}$ is the data.

Likelihood function

Let's suppose that we have no prior knowledge over $\boldsymbol{\theta}$, so we'll drop the prior and focus specifically on the likelihood:

$$\mathcal{L}(\boldsymbol{\theta}) = p(\mathbf{X}|\theta)$$

How would we calculate this?

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= p(\mathbf{x_1}|\boldsymbol{\theta}) \times p(\mathbf{x_2}|\boldsymbol{\theta}) \times \ldots \times p(\mathbf{x_n}|\boldsymbol{\theta}) \\ &= \prod_{i=1 \ldots N} p(\mathbf{x}_i|\boldsymbol{\theta})\end{aligned}$$

i.e. the product of the probability of each observations within $\mathbf{X}$, given $\boldsymbol{\theta}$.

What does this assume?

Why is the likelihood function useful?

Suppose we have two alternative values of $\boldsymbol{\theta}$: $\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}$. We can calculate the likelihood *ratio* (LR) of these two possible parameter values:

$$LR = \frac{\mathcal{L}(\boldsymbol{\theta}^{(1)})}{\mathcal{L}(\boldsymbol{\theta}^{(2)})}$$

If $LR > 1$, which parameter value would we pick?

Maximum likelihood estimation

We can generalise this for all possible values of θ:

$$\arg\max_{\boldsymbol{\theta} \in \Theta} \mathcal{L}(\boldsymbol{\theta}) = \prod_{i=1 \ldots N} p(\mathbf{x}_i | \boldsymbol{\theta})$$

i.e., from the set of all possible parameter values Θ, find the parameter value that maximises the likelihood function.

Hence, **maximum** likelihood estimation.

- How we calculate $p(\mathbf{x}_i | \boldsymbol{\theta})$ will depend on the functional form of the underlying distribution
- We'll explore this specifically with respect to logistic regression later on today

Why is this useful?

Numeric overflow

Multiplying many small numbers means we soon lose the power to calculate them precisely

- `R` double-precision numbers range from 2×10^{-308} to 2×10^{308}
- If 400 observations have $p_{\boldsymbol{\theta}} = 0.01$, $\mathcal{L}(\boldsymbol{\theta})$ will be outside the computable range

What if we take the log?

- The log function is strictly increasing
- $Log(a \times b) = Log(a) + Log(b)$ so we can simply add the values

With the logged likelihood function we do not have the problem of numeric overflow!

Negative log-likelihood

One final step that we can take is to calculate the *negative* log-likelihood:

- We then *minimise* the negative log-likelihood
- We typically want to minimise rather than maximise because many of our procedures for optimisation are based on the former
- But, broadly, this is just semantics:
    - Minimising the negative log-likelihood is the same as maximising the log-likelihood

Logistic regression

Logistic regression:

- Allows us to estimate β parameters when we have a binary outcome variable
- More broadly, it is a **binary classification** algorithm – what is the probability that $y_i = 1$ given a vector of features $\mathbf{x_i}$?

We can write the logistic regression function as,

$$f_{\theta,b}(\mathbf{X}) = \frac{1}{1 + e^{-(\theta\mathbf{X}+b)}}.$$

The goal is to find the *best* values of $\boldsymbol{\theta}$ and b that "explains" the data

- Let's subsume within $\boldsymbol{\theta}$ s.t. $\boldsymbol{\theta} = \{b, \theta_1, \cdots, \theta_k\}$

MLE of logistic regression

For a given vector of scalar values $\boldsymbol{\theta}$, we can ask what the likelihood of the data is given those values

How do we construct this?

- If $y_i = 1$, we want the $f_{\boldsymbol{\theta}}(\mathbf{x_i})$
- But if $y_i = 0$ we want the inverse, i.e. $(1 - f_{\boldsymbol{\theta}}(\mathbf{x_i}))$
- We can combine these two using a mathematical "logic gate":

$$\mathcal{L}_{\boldsymbol{\theta}} = f_{\boldsymbol{\theta}}(\mathbf{X})^{\boldsymbol{y}} \times (1 - f_{\boldsymbol{\theta}}(\mathbf{X}))^{(1-\boldsymbol{y})},$$

since when $y_i = 0, x^{y_i} = 1$ and $x^{(1-y_i)} = x$, and vice versa.

Simplifying, since $f_{\boldsymbol{\theta}}(\boldsymbol{x_i}) = \hat{y}_i$:

$$\mathcal{L}_{\boldsymbol{\theta}} = \hat{\boldsymbol{y}}^y(1 - = \hat{\boldsymbol{y}}_i)^{1-y}$$

MLE optimization

We can then apply our "tricks" to make the computation easier:

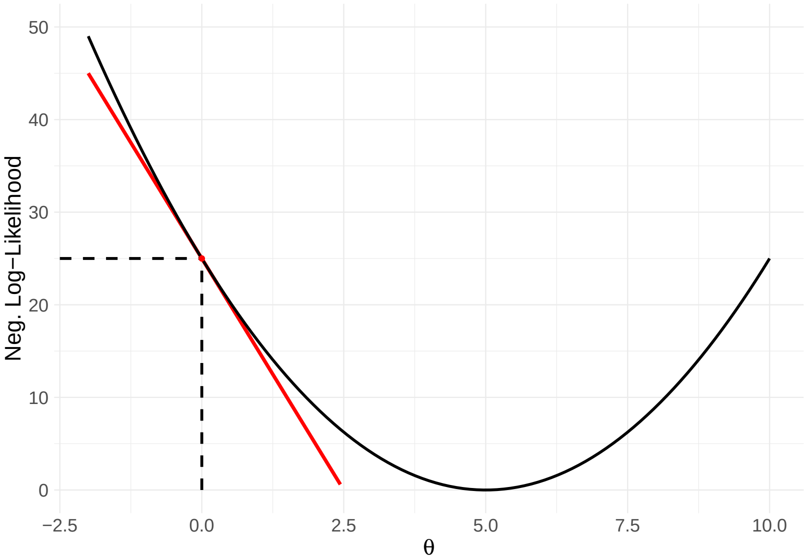
$$-Log(\mathcal{L}_{\boldsymbol{\theta}}) = -\sum_{i=1}^{N} Log\left(\mathcal{L}_{\boldsymbol{\theta}}(\mathbf{x_i})\right),$$

with the goal of minimising this quantity through choosing $\boldsymbol{\theta}$.

How exactly do we minimize this function?

- Unlike OLS, where there is a closed form solution, it is not possible to analytically minimize the negative log-likelihood of the logistic regression
- We therefore have to use computation to iterate through values of $\boldsymbol{\theta}$ to approximate the minima

Minimising the negative log-likelihood in one dimension



Gradient descent

To find the minimum of the negative log-likelihood we:

1. Choose a value for the starting parameter θ
2. Calculate the slope of the function at that point
3. Adjust our value of θ in the opposite direction to the slope coefficient's sign
4. Recalculate the slope, and repeat 2-4

We can generalise this to $\boldsymbol{\theta}$:

- Let $Q(\boldsymbol{\theta})$ be the negative log likelihood function
- Calculate the **gradient** vector of the function in k-dimensions
- Adjust each parameter $\theta_k \in \boldsymbol{\theta}$ by the negative of the corresponding element of the gradient

$$\theta_k = \theta_k - \lambda \frac{\partial Q(\boldsymbol{\theta})}{\partial \theta_k}$$

Logistic regression gradient

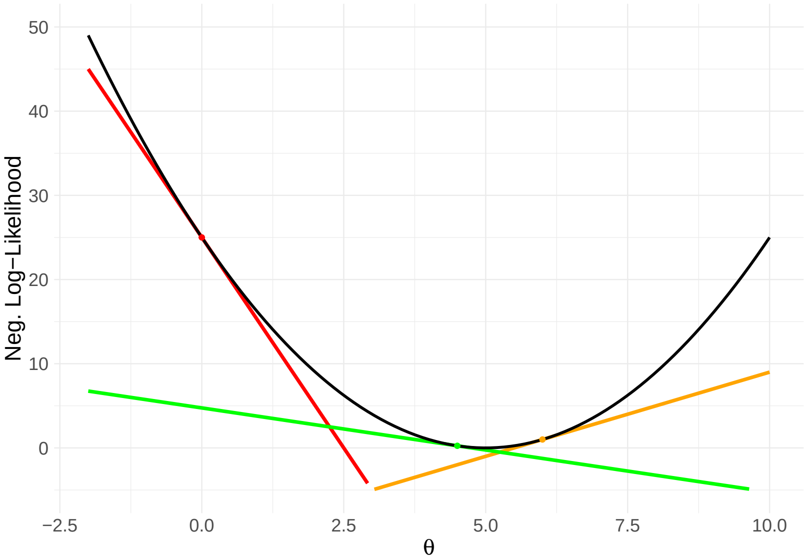
The partial derivative for any predictor $\boldsymbol{x}^{(j)}$ for the *logistic* cost function is:

$$\frac{\partial Q^{\text{Logit}}}{\partial \theta_k} = (f_{\theta_k}(\boldsymbol{X}) - \boldsymbol{y})\boldsymbol{x}^{(k)}$$

Hence the gradient of the function's curve for any vector of logistic parameters $\boldsymbol{\theta}$ can be described as:

$$\boldsymbol{\nabla} = \begin{bmatrix} \frac{\partial Q^{\text{Logit}}(\boldsymbol{\theta})}{\partial \theta_1} \\ \frac{\partial Q^{\text{Logit}}(\boldsymbol{\theta})}{\partial \theta_2} \\ \vdots \\ \frac{\partial Q^{\text{Logit}}(\boldsymbol{\theta})}{\partial \theta_k} \end{bmatrix}$$

Progression of the descent algorithm



Learning rate

How do we know how much to adjust the parameters by?

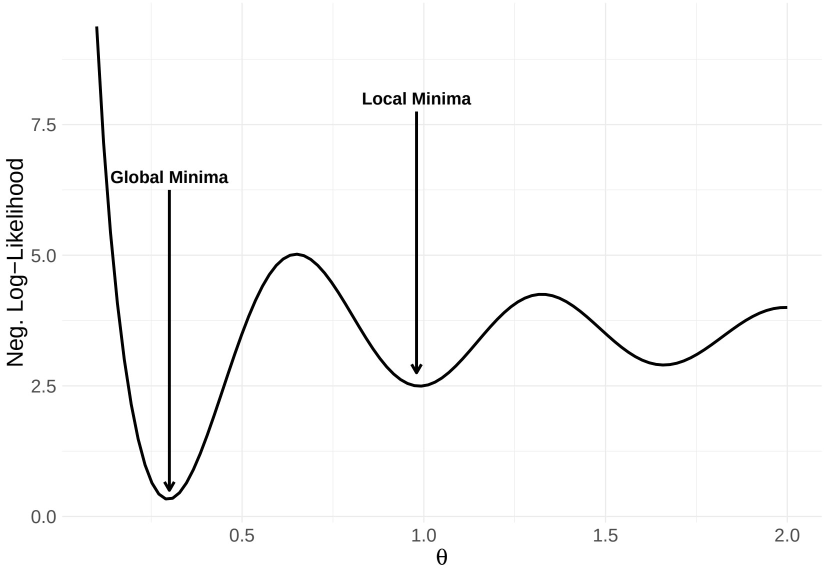
- Intuitively, when our gradient is large we want to make big adjustments because we are far away from the minimum
- As we get closer, i.e. the gradient is smaller, we want to finetune our adjustments and make smaller steps

So we can scale our adjustment by the size of the gradient

- Let's call this hyperparameter the **learning rate** (λ)
- $\boldsymbol{\theta}_{\text{New}} = \boldsymbol{\theta} - \boldsymbol{\nabla}\lambda$

The choice of λ is down to the researcher:

- Overly-large values will prevent minimisation
- Overly small values may take too long, or risk converging on *local* minima



Local Minima
Global Minima
Neg. Log−Likelihood
7.5
5.0
2.5
0.0
0.5
1.0
1.5
2.0
θ

Stochastic gradient descent

Gradient descent can be **expensive**:

- We have to evaluate all rows in our training data before making any updates to the parameters
- If we have lots of observations
  1. Each calculation takes a long time
  2. Take many iterations to optimise
- Instead we can use **stochastic gradient descent** (SGD)
  - Inspect the loss of each observation (or a random subset) individually
  - Update the coefficients based on each observation

Stochastic gradient descent

Under GD, for each iteration:

$$\theta_k \leftarrow \theta_k + \lambda \sum_{i=1}^{N} (y_i - f_{\theta_k}(\boldsymbol{x_i}))\boldsymbol{x_i}^{(k)}$$

Under SGD, for each iteration:

$$\theta_k \leftarrow \sum_{i=1}^{N} \theta_k + \lambda (y_i - f_{\theta_k}(\boldsymbol{x_i}))\boldsymbol{x_i}^{(k)}$$

- SGD typically converges a lot faster than GD
  - Every iteration we make N small changes to the parameter estimate
  - Computationally more efficient (we'll cover this more later in the week)
  - At the cost of some additional noise in the optimisation process

Coding workshop: writing our own logistic regression classifier