MA5832: Data Mining & Machine Learning

Collaborate Week 2: Optimisation & Probability

Martha Cooper, PhD

JCU Masters of Data Science

2021-01-21

Housekeeping

- Collaborates = Thursdays 6-7:30pm

For my Collaborate Sessions, you can get the slides & R code for each week on Github:

https://github.com/MarthaCooper/MA8532

Today's Goals

- Optimisation
  - Gradient Descent
- Probability

Optimisation

Formal Definition of Optimisation

Optimisation is the selection of a best elements from an available set of alternatives. The mathematical notation for finding $\boldsymbol{x}^*$ which *minimises* the *loss function* $\boldsymbol{f}(\boldsymbol{x})$ is

$$x^* = arg \min_{x^*} f(x)$$

- We are going to learn an unconstrained optimisation algorithm called gradient descent.
- Extension: look up Newton's method & more...

Optimisation in the context of machine learning

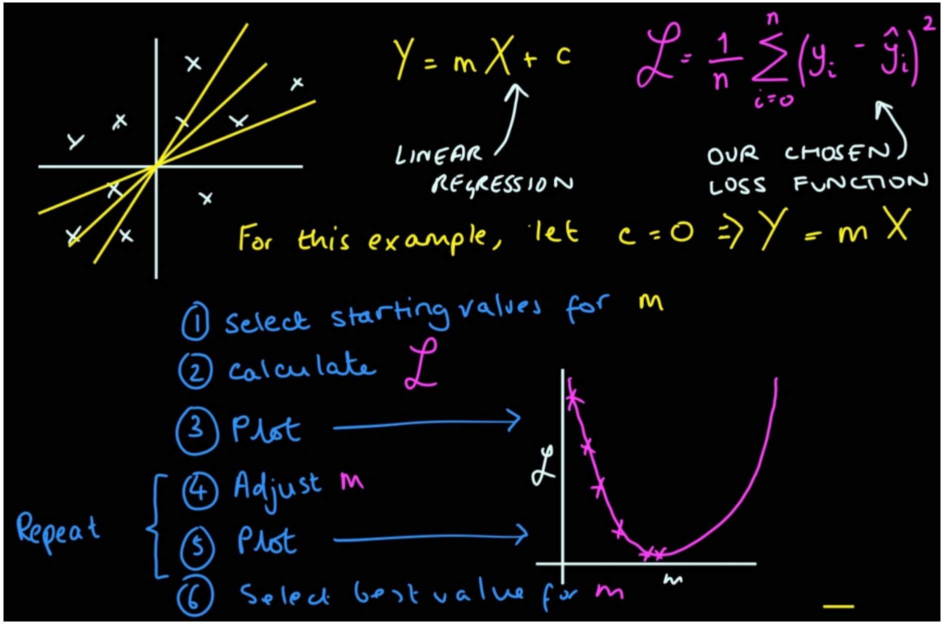
1. Define a Loss Function
2. Use Optimisation to minimise the Loss Function

e.g. Gradient Descent

Important for decision trees, SVM, neural networks

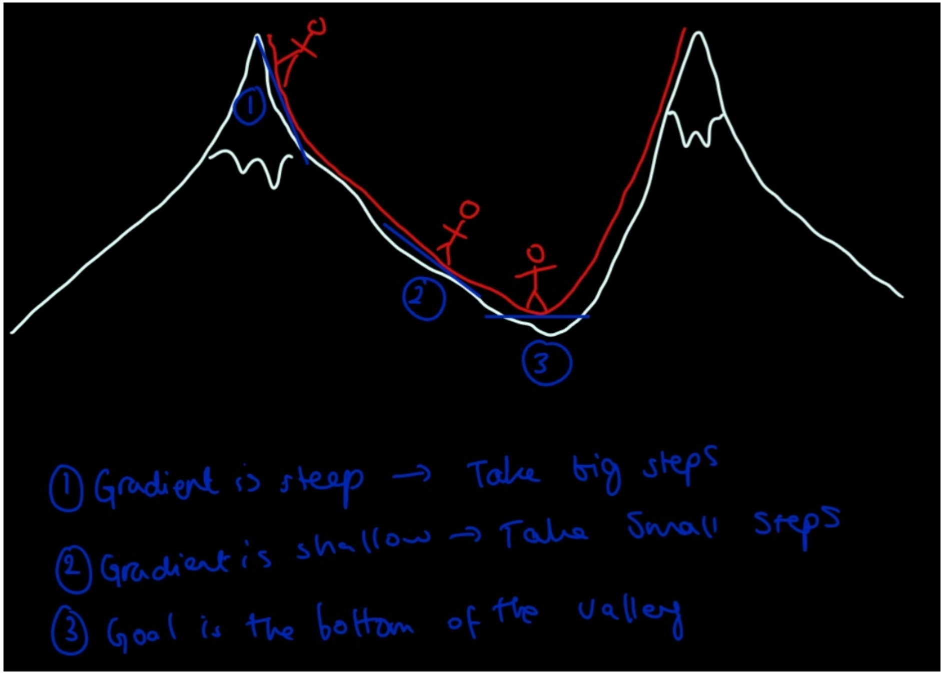
Why use optimisation?

- Can be used to estimate parameters in statistical & machine learning models
- We will use an example - Linear Regression.
- Note: There are many ways to estimate parameters for Linear Regression and other methods for estimated linear regression parameters are generally preferred over optimisation, but I found this a useful way learn about optimisation & gradient descent intuitively…



Optimisation with Gradient Descent

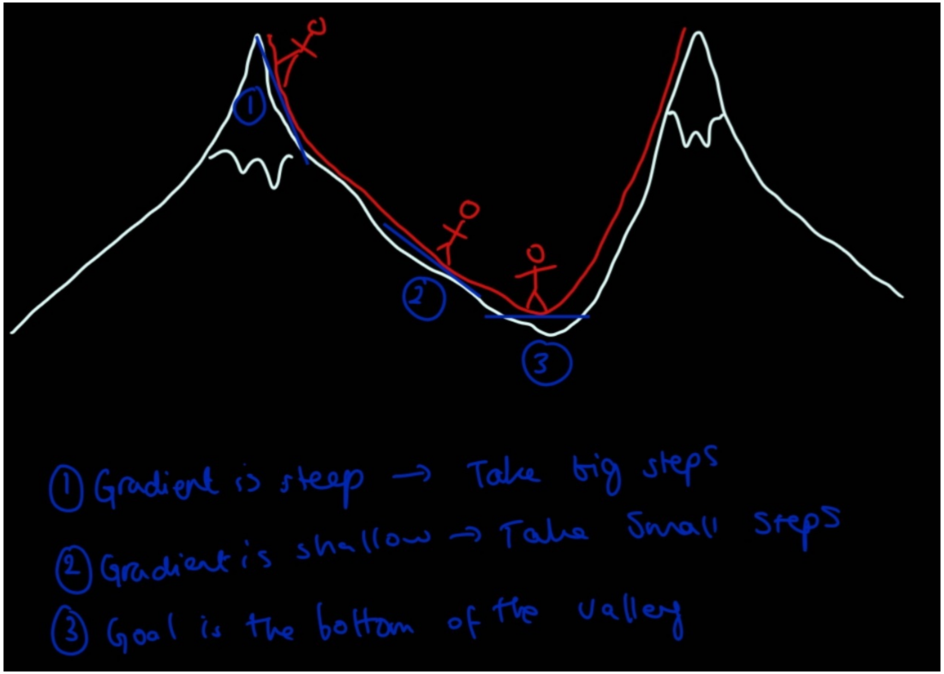
Gradient Descent is an iterative algorithm to find the minimum of a function.



1. ... how to find the gradient?
2. ... how to find the speed?

Optimisation with Gradient Descent

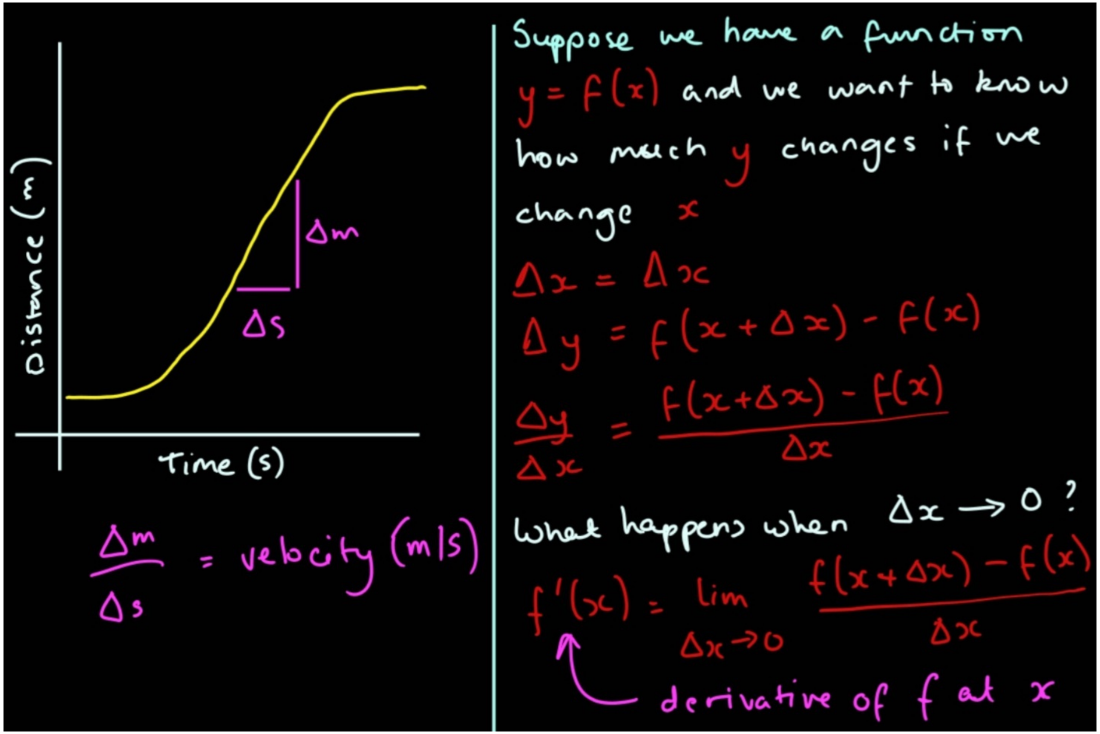
Gradient Descent is an iterative algorithm to find the minimum of a function.



1. ... how to find the gradient? First derivative
2. ... how to find the speed? Learning Rate

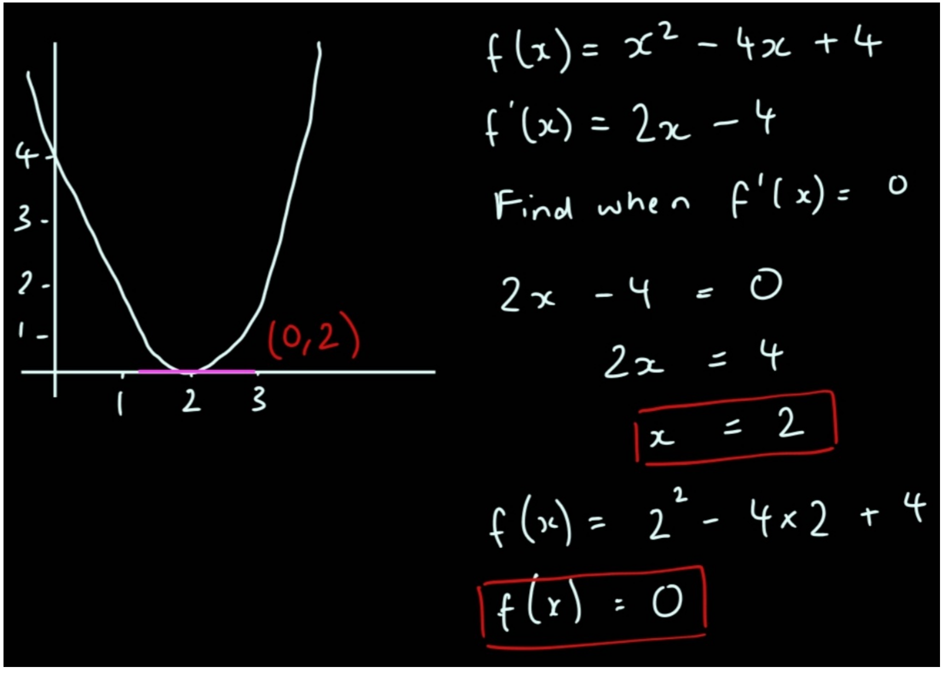
The First Derivative - Revision

- Rate of Change
- The derivative of a function $y = f(x)$ of a variable is a measure of the rate at which the value y changes with respect to the change in x



Using The First Derivative for Optimisation

The first derivative is useful for optimisation because it computes the direction of the slope of the function and we can use it to find the minimum point of the function.



- x < 2 then f'(x) < 0
- x > 2 the f'(x) > 0

Differentiation Rules

Basic Derivatives Rules

Constant Rule: $\frac{d}{dx}(c) = 0$

Constant Multiple Rule: $\frac{d}{dx}[cf(x)] = cf'(x)$

Power Rule: $\frac{d}{dx}(x^n) = nx^{n-1}$

Sum Rule: $\frac{d}{dx}[f(x) + g(x)] = f'(x) + g'(x)$

Difference Rule: $\frac{d}{dx}[f(x) - g(x)] = f'(x) - g'(x)$

Product Rule: $\frac{d}{dx}[f(x)g(x)] = f(x)g'(x) + g(x)f'(x)$

Quotient Rule: $\frac{d}{dx}\left[\frac{f(x)}{g(x)}\right] = \frac{g(x)f'(x) - f(x)g'(x)}{[g(x)]^2}$

Chain Rule: $\frac{d}{dx}f(g(x)) = f'(g(x))g'(x)$

onlinemathlearning

Extension = multivariate functions/partial derivative

Gradient Descent Algorithm

1. Define a starting point for the variable(s) we want to optimise, and define a learning rate. *Given a starting point* $\boldsymbol{x}^{(k)}, \boldsymbol{k} = \boldsymbol{0}$. *Let the Learning Rate be defined as* $\boldsymbol{\alpha}_{\boldsymbol{k}}$.

2. Calculate the (partial) derivative of the loss function. *Find the gradient of* $f'(x)^{(k)}$

3. Update our current value of $x^{(k)}$ to $x^{(k+1)}$ using the equation $x^{(k+1)} = x^{(k)} - \alpha_k f'(x^{(k)})$

4. Repeat steps 2 - 4 a large number of times.

Applying gradient descent to linear regression (intecept fixed at 0)

$Y = mX + c$. Let $c = 0$, so $Y = mX$.

$$\mathcal{L} = \frac{1}{n}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=0}^{n}(y_i - mX)^2$$

1. Let $m = 0$. Let $L = 0.001$
2. Calculate derivative of loss function

$$D_m = \frac{-2}{n}\sum_{i=0}^{n} x_i(y_i - \hat{y}_i)$$

3. Update current value of m using:

$$m = m - L \times D_m$$

4. Repeat until the loss function is minimised

Gradient Descent in R

```
# Goal is to estimate m in a simple linear regression y = mX using Gradi
# Won't all fit on this slide so check the .rmd!

lm_gd<-function(x, # vector of x values
                y, # vector of y values
                m0, # starting point for m
                step.size=0.05, # learning rate (equivalent to alpha in
                max.iter=100, # repeat process 100 times (higher iterati
                changes=0.001){# if the gradient is smaller than the thi
                               # go back and re-run the function before

  # Store the values of m across number of iterations
    m<-matrix(0, ncol=1, nrow=max.iter) # matrix to store parameter esti
    d_m <- matrix(0, ncol = 1, nrow=max.iter) # matrix to store gradient

  # Step 1 in Gradient Descent method
    m[1,]<-m0 #set first variable to 0

    for(i in 1:(max.iter-1)){
      yhat <- m[i,1]*x #calculate yhats for all xs and m0

      # Step 2: calculate the gradient of the loss function using the de
      #the loss function with respect to m
      d_m[i,1] <- -2*mean(x*(y-yhat))
```

Application in R

```
#simulate some data with c = 0
x <- rnorm(40, mean = 0, sd = 1)
y <- 3*x+0+rnorm(40,0,2)
plot(x,y) #plot

l <- lm_gd(x,y,m0=0, step.size = 0.001, max.iter = 10000) #lm with gd

plot(l$d_m, l$m) #plot results
tail(l) #view results

plot(x,y) #plot
abline(0, 3.18) #fit linear regression line

#calculate the loss function for each value of m

l$loss <- 0
for(i in 1:nrow(l)){
  l$loss[i] <- mean((y-(l$m[i]*x))^2)
}

plot(l$m, l$loss)
```

Extensions

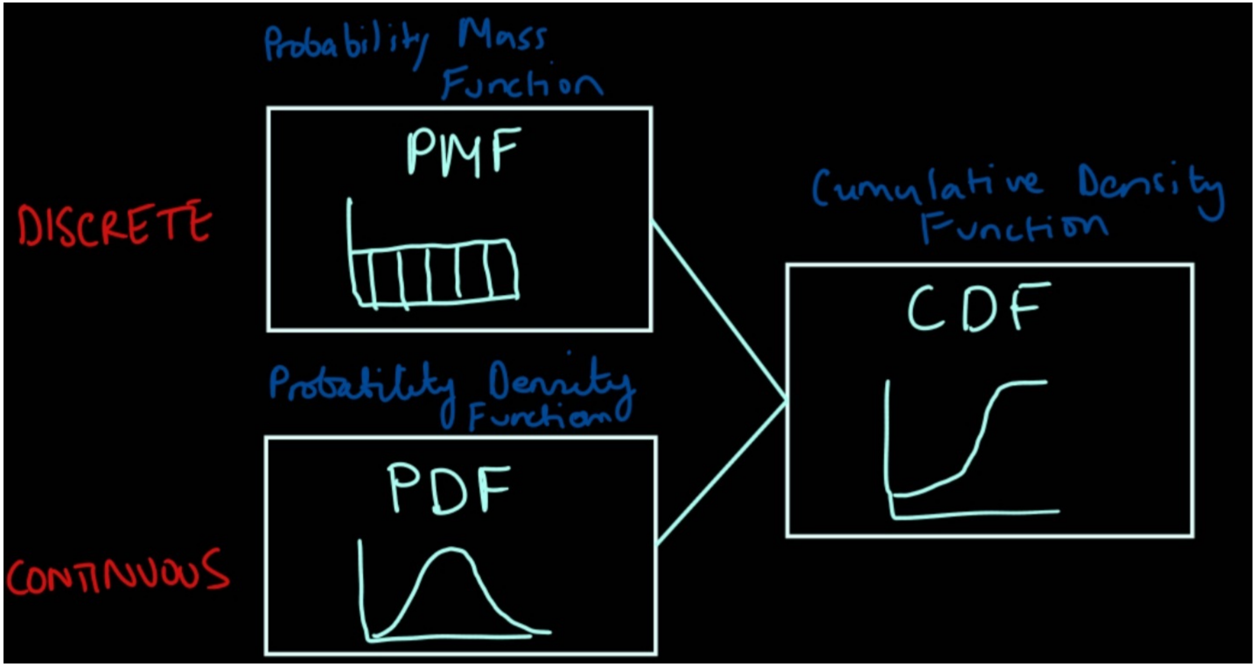
- What happens when you change the step size?
- How do you also choose the best value for the intercept? (clue: partial derivatives)

Probability

Probability Review

- A discrete random variable has a countable number of possible values.
  - Discrete uniform distributions
  - Binomial distribution
  - Poisson distribution
- A continuous random can take on any value within a range of values.
  - Normal distribution
  - t-distribution
  - uniform distribution

Probability Distribution Functions



Extra reading/watching

Optimisation

- Essence of Calculus by 3Blue1Brown
- Towards Data Science Demystifying Optimisation

Probability

- Chapter 3 Deep Learning by Goodfellow, Bengio & Courville

References

- Towards Data Science

Slides

- xaringhan, xaringanthemer, remark.js, knitr, R Markdown