Modern Computational Statistics

Richard Wilkinson

School of Mathematics and Statistics
University of Sheffield

September 18, 2015

Introduction

The explosion in computer power and computational techniques has led to huge changes in statistics/machine learning.

- HPC
- Monte Carlo methods
- Probabilistic programming, e.g., STAN, WinBUGS,

Models can now be fitted and used in a way that couldn't have been conceived of before.

- Model complexity
- Big data
- Enabled the increasing dominance of Bayesian methods

Aim of this session is not to teach algorithmic details, but describe what is available for each type of problem.

Recap: Monte Carlo integration

Suppose we are interested in the integral

$$I = \mathbb{E}(g(X)) = \int g(x)f(x)\mathrm{dx}$$

e.g. $\mathbb{P}(A|D) = \int \mathbb{I}_{\theta \in A}\pi(\theta|D)\mathrm{d}\theta, \qquad \mathbb{E}(\mathrm{T}|\mathrm{D}) = \int \mathrm{T}\pi(\mathrm{T}|\theta)\pi(\theta|\mathrm{D})\mathrm{d}\theta$

Recap: Monte Carlo integration

Suppose we are interested in the integral

$$I = \mathbb{E}(g(X)) = \int g(x)f(x)\mathrm{dx}$$

e.g. $\mathbb{P}(A|D) = \int \mathbb{I}_{\theta \in A}\pi(\theta|D)\mathrm{d}\theta, \qquad \mathbb{E}(\mathrm{T}|\mathrm{D}) = \int \mathrm{T}\pi(\mathrm{T}|\theta)\pi(\theta|\mathrm{D})\mathrm{d}\theta$

Let $X_1, X_2, \ldots, X_n$ be independent random variables with pdf $f(x)$. Let

$$\hat{I}_n = \frac{1}{n}\sum_{i=1}^{n} g(X_i). \tag{1}$$

The main idea in Monte Carlo integration is to approximate I by $\hat{I}_n$

(1) $\hat{I}_n$ is an unbiased estimator of I.
(2) $\hat{I}_n$ converges to I as $n \to \infty$.
(3) The central limit theorem tells us the rate of convergence of $\hat{I}_n$:

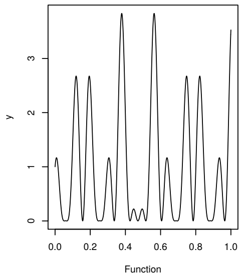
$$\hat{I}_n \sim N(I, \frac{\sigma^2}{n}) \text{ where } \sigma^2 = \mathbb{V}\mathrm{ar}[g(X)]$$

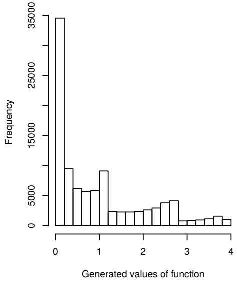
Monte Carlo Example

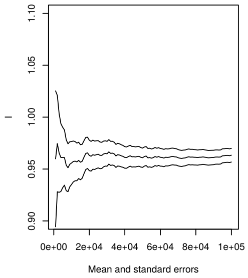
Consider the integral $\int_0^1 h(x)f(x)\mathrm{d}x$ where

$$h(x) = [\cos(50x) + \sin(20x)]^2 \qquad f(x) = \begin{cases} 1 \text{ if } x \in [0,1] \\ 0 \text{ otherwise} \end{cases}$$

Generate $X_1, \ldots, X_n$ from $U[0,1]$ and estimate with $\hat{I}_n = \frac{1}{n}\sum h(X_i)$.





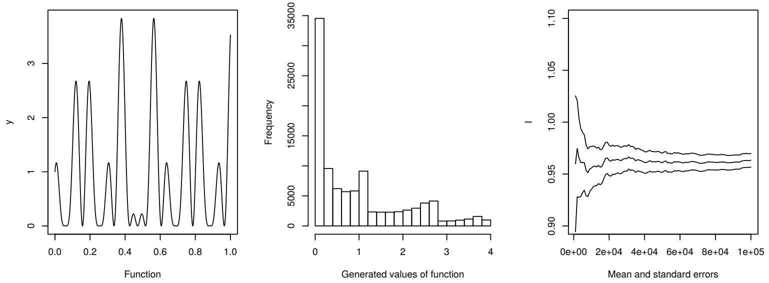


Monte Carlo Example

Consider the integral $\int_0^1 h(x) f(x) \mathrm{d}x$ where

$$h(x) = [\cos(50x) + \sin(20x)]^2 \qquad f(x) = \begin{cases} 1 \text{ if } x \in [0,1] \\ 0 \text{ otherwise} \end{cases}$$

Generate $X_1, \ldots, X_n$ from $U[0,1]$ and estimate with $\hat{I}_n = \frac{1}{n} \sum h(X_i)$.



There are many ways of reducing the variance of the estimator.
The difficulty is in generating samples from $f(x)$ particularly when $f(x) = \pi(x|D)$

Bayesian inference

The Bayesian approach to statistics is beautifully simple

- Uncertainty is represented by probability
  - Explain the difference between likelihood, confidence, probability and a p-value.
- Bayes theorem used to combine probabilities

$$\pi(X|D) = \frac{\pi(X)\pi(D|X)}{\pi(D)}$$

$$\text{posterior} \propto \text{prior} \times \text{likelihood}$$

Bayesian inference

The Bayesian approach to statistics is beautifully simple

- Uncertainty is represented by probability
  - Explain the difference between likelihood, confidence, probability and a p-value.
- Bayes theorem used to combine probabilities

$$\pi(X|D) = \frac{\pi(X)\pi(D|X)}{\pi(D)}$$

$$\text{posterior} \propto \text{prior} \times \text{likelihood}$$

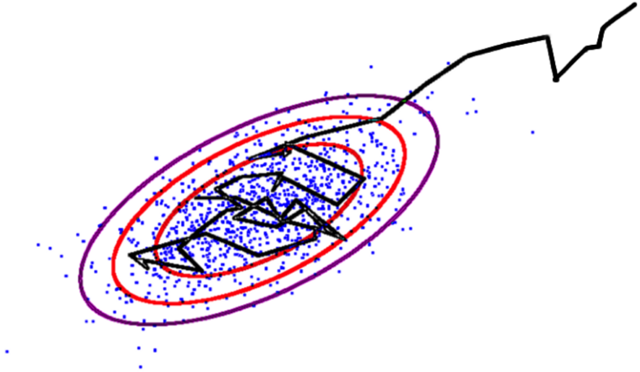
However, while philosophically this is simple and the same in **every** problem, computation is hard.

For most models, we have to resort to approximation, e.g. Monte Carlo, to compute the posterior.

MCMC

Markov chain Monte Carlo (MCMC) is a class of algorithms for sampling from a distribution, e.g., a posterior distribution.

- Construct a Markov chain $X_1, X_2, \ldots$ such that samples from this chain are samples from the distribution of interest, e.g., $\pi(X|D)$



Metropolis-Hastings Algorithm

To sample from $\pi(x|D)$

Metropolis-Hastings Algorithm

1. Suppose at time t, we have $X_t = x$. Propose a candidate value y from proposal distribution $q(x, y)$.
2. Calculate the acceptance probability $\alpha(x, y)$
$$\alpha(x,y) = \min\left(1, \frac{\pi(D|y)\pi(y)q(y,x)}{\pi(D|x)\pi(x)q(x,y)}\right)$$
3. $$\text{Set } X_{t+1} = \begin{cases} y & \text{with probability } \alpha(x,y) \\ x & \text{with probability } 1-\alpha(x,y) \end{cases}$$

$q(x, y)$ must be easy to sample from and obey some simple rules.

- random walks are common choices

Acceptance probability α converts the Markov chain from the wrong distribution, to the desired distribution.

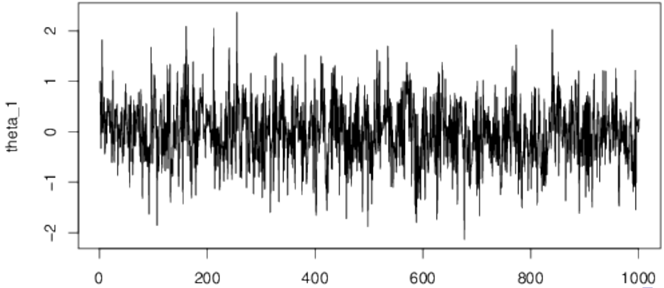
How to spot failure

Theory says samples from MCMC, $X_1, X_2, \ldots$ converge to a sample from $\pi(X|D)$ regardless of choice of $q^{\dagger}$.

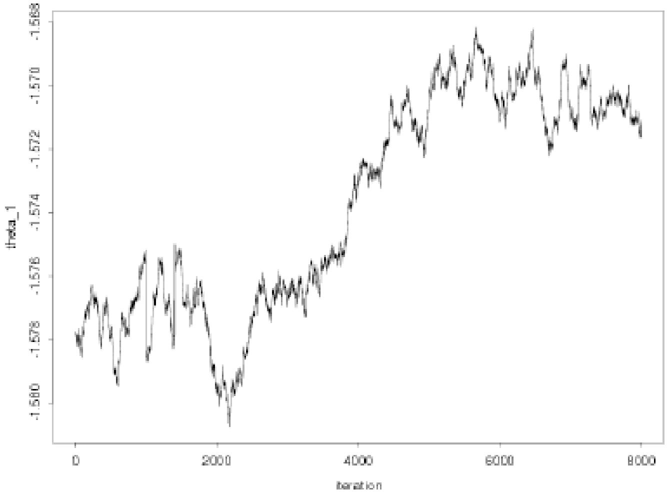
- We must check convergence
  - burn in
- And mixing (has the chain explored all of space)
  - thinning

A poor choice of q will lead to nonsense. Aim for an acceptance rate of $\sim 20\%$

Trace and autocorrelation plots are useful.

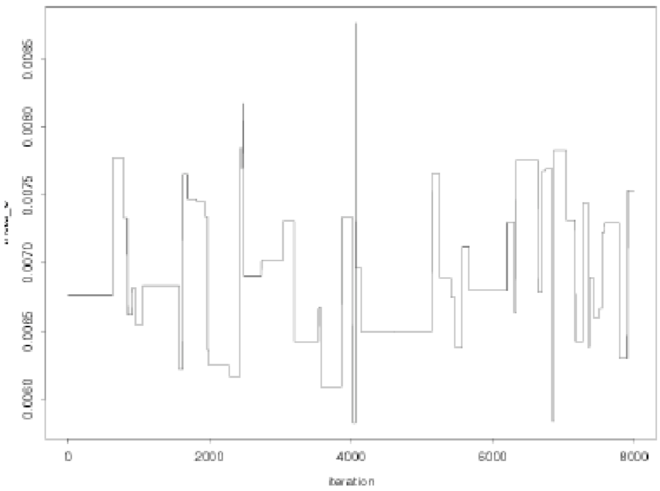


MCMC Problems - Example 1



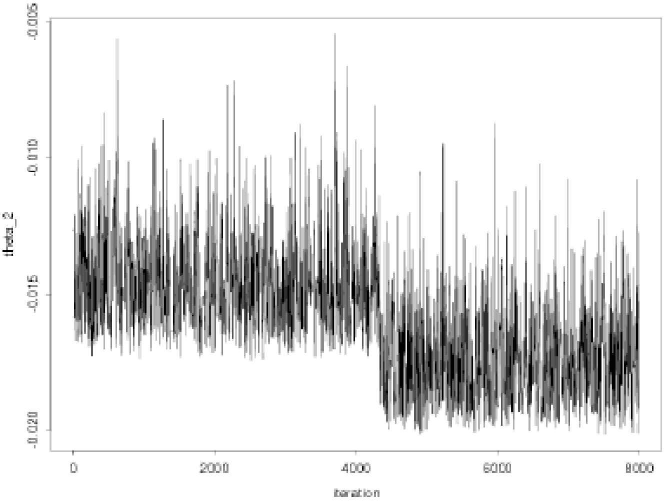
Too small a step size in q

MCMC Problems - Example 2



Low acceptance rate - try smaller moves in q, and/or different choice

MCMC Problems - Example 3



Bi-modal posterior with poor mixing - try a boutique choice for q

Advanced MCMC

MCMC allows for an almost arbitrary choice of proposal $q(x, y)$.

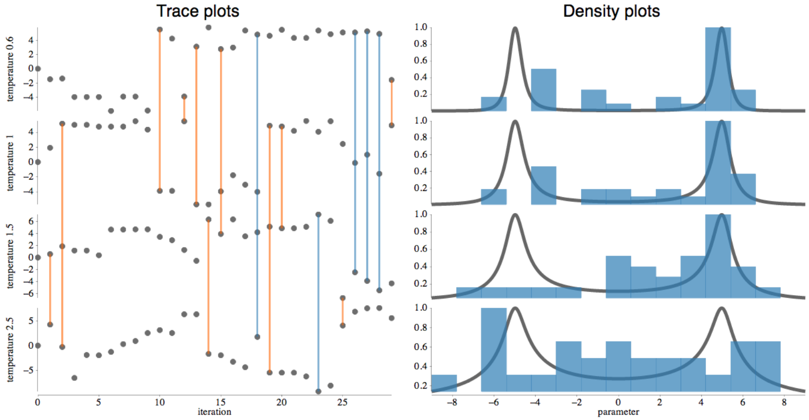
A large volume of work exists on good chocies of q

- Gibbs sampling
  - WinBUGS
- Adaptive MCMC -$y \sim q(x, \cdot) = N(x, \Sigma)$ automatically tune Σ
- Hybrid/Hamiltonian Monte Carlo
  - Introduce dynamics - requires derivatives $\frac{d}{dx} \log(\pi(D|x)\pi(x))$
  - Good for strange shape likelihood functions
  - STAN
- Slice sampling
- Tempering
  - Works well for multimodal posteriors
- ...

Plus combinations of all of the above

Parallel tempering

Run multiple MCMC chains targetting $\pi(x|D)^{p_i}$ for $p_i \leq 1$



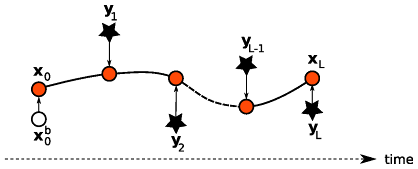
http://www.youtube.com/watch?v=J6FrNf5__G0

Data assimilation

Data assimilation

Assume we have a time structured problem

$$x_{t+1} = f(x_t) + u_t$$
$$y_t = g(x_t) + v_t$$



If f and g are linear functions, and u_t and v_t are Gaussian, the Kalman filter (KF) gives us

$$\pi(x_{1:t}|y_{1:t})$$

For non-linear problems, the ensemble KF or unscented KF approximate the filtering distributions using a Gaussian approximation.

Particle filter/SMC

Represent a distribution by a set of weighted particles $\{x_i, w_i\}_{i=1}^n$

$$\pi(x) \approx \sum w_i \delta_{x_i}(x)$$

The particle filter builds a (non-Gaussian approximation) to $\pi(x_t|y_{1:t})$

- Start: $\{x_i^{(t)}, w_i^(t)\}_{i=1}^n \approx \pi(x_t|y_{1:t})$
- Propagate: $x_i^{(t+1)} = f(x_i^{(t)}) + u_t$
- Reweight: $w_i^{(t+1)} \propto \pi(y_{t+1}|x_i^{(t+1)})$
- Resample if necessary.

Particle filter/SMC

Represent a distribution by a set of weighted particles $\{x_i, w_i\}_{i=1}^n$

$$\pi(x) \approx \sum w_i \delta_{x_i}(x)$$

The particle filter builds a (non-Gaussian approximation) to $\pi(x_t|y_{1:t})$

- Start: $\{x_i^{(t)}, w_i^(t)\}_{i=1}^n \approx \pi(x_t|y_{1:t})$
- Propagate: $x_i^{(t+1)} = f(x_i^{(t)}) + u_t$
- Reweight: $w_i^{(t+1)} \propto \pi(y_{t+1}|x_i^{(t+1)})$
- Resample if necessary.

Sequential Monte Carlo (SMC) adapts the PF to sample from $\pi(\theta|D)$

- Sample from $\pi_1(\theta|D)$ (something easy, e.g. $\pi(\theta)$)
- Reweight and propagate θ particles to sample from

$$\pi_2(\theta|D)$$
$$\ldots$$
$$\pi_T(\theta|D) = \pi(\theta|D)$$

The number of particles required depends upon $\dim(x)$ and $T =$length of time series.

Degeneracy

For hard problems, we can quickly find degeneracy

- A few particles have all the weight

We can try to avoid this using

- Importance sampling and clever propagation proposals
- Resampling the particles

Degeneracy

For hard problems, we can quickly find degeneracy

- A few particles have all the weight

We can try to avoid this using

- Importance sampling and clever propagation proposals
- Resampling the particles

Solving the joint calibration and filtering problem:

$$\begin{array}{cl} x_{t+1} & = f_\theta(x_t) + u_t \\ y_t & = g_\theta(x_t) + v_t \end{array} \implies \begin{array}{l} \pi(x_{1:t}, \theta | y_{1:t}) \\ \pi(\theta | y_{1:t}) \end{array}$$

is much harder.

- Pseudo-marginal methods such as Particle MCMC, SMC^2

ABC

Intractability

$$\pi(\theta|D) = \frac{\pi(D|\theta)\pi(\theta)}{\pi(D)}$$

- usual intractability in Bayesian inference is not knowing $\pi(D)$.
- a problem is doubly intractable if $\pi(D|\theta) = c_\theta p(D|\theta)$ with c_θ unknown (cf Murray, Ghahramani and MacKay 2006)
- a problem is completely intractable if $\pi(D|\theta)$ is unknown and can't be evaluated (unknown is subjective). I.e., if the analytic distribution of the simulator, $f(\theta)$, run at θ is unknown.

Completely intractable models are where we need to resort to ABC methods

Approximate Bayesian Computation (ABC)

Given a complex simulator for which we can't calculate the likelihood function - how do we do inference?

Approximate Bayesian Computation (ABC)

Given a complex simulator for which we can't calculate the likelihood function - how do we do inference?

If its cheap to simulate, then ABC (approximate Bayesian computation)is one of the few approaches we can use.

ABC algorithms are a collection of Monte Carlo methods used for calibrating simulators

- they do not require explicit knowledge of the likelihood function
- inference is done using simulation from the model (they are 'likelihood-free').

Approximate Bayesian Computation (ABC)

Given a complex simulator for which we can't calculate the likelihood function - how do we do inference?

If its cheap to simulate, then ABC (approximate Bayesian computation)is one of the few approaches we can use.

ABC algorithms are a collection of Monte Carlo methods used for calibrating simulators

- they do not require explicit knowledge of the likelihood function
- inference is done using simulation from the model (they are 'likelihood-free').

ABC methods are primarily popular in biological disciplines

- Simple and intuitive to implement
- Embarrassingly parallelizable
- Can usually be applied

Rejection ABC

Sample from

$$\pi(\theta|D) \propto \pi(\theta)\pi(D|\theta)$$

where $\pi(D|\theta)$ is the likelihood corresponding to a stochastic simulator $f(\theta)$

Uniform Rejection Algorithm

- Draw θ from $\pi(\theta)$
- Simulate $X \sim f(\theta)$
- Accept θ if $\rho(D, X) \leq \epsilon$

Rejection ABC

Sample from

$$\pi(\theta|D) \propto \pi(\theta)\pi(D|\theta)$$

where $\pi(D|\theta)$ is the likelihood corresponding to a stochastic simulator $f(\theta)$

Uniform Rejection Algorithm

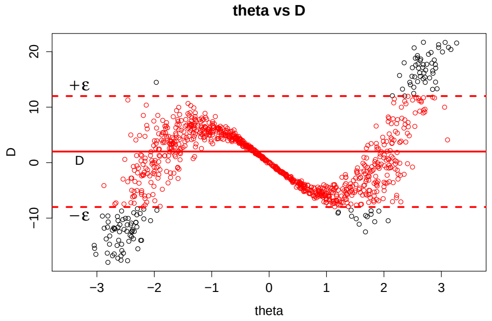
- Draw θ from $\pi(\theta)$
- Simulate $X \sim f(\theta)$
- Accept θ if $\rho(D,X) \leq \epsilon$

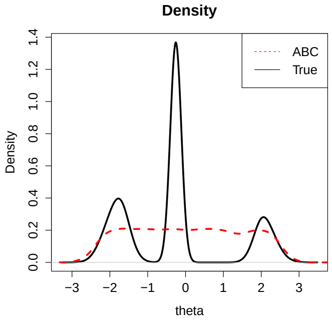
This generates observations from $\pi(\theta \mid \rho(D,X) < \epsilon)$:

- As $\epsilon \to \infty$, we get observations from the prior, $\pi(\theta)$.
- If $\epsilon = 0$, we generate observations from $\pi(\theta \mid D)$.

ϵ reflects the tension between computability and accuracy.

$\epsilon = 10$

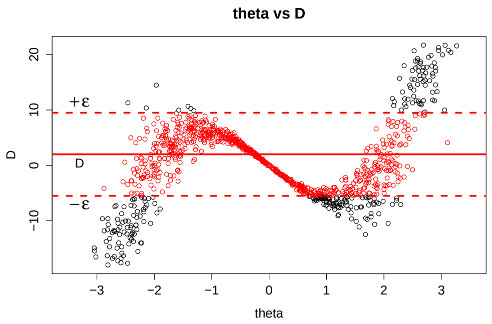


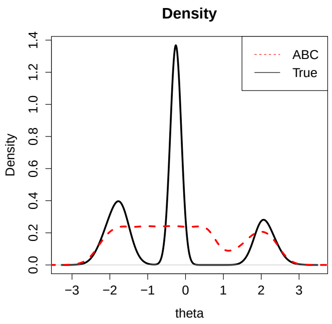


$$\theta \sim U[-10, 10], \qquad X \sim N(2(\theta + 2)\theta(\theta - 2), 0.1 + \theta^2)$$

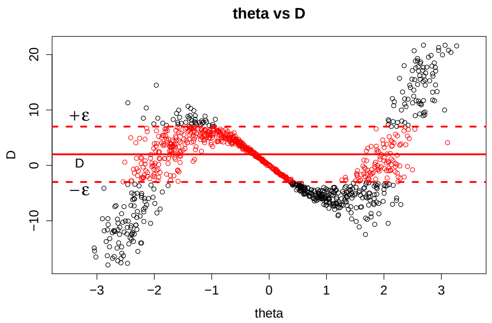
$$\rho(D, X) = |D - X|, \qquad D = 2$$

$\epsilon = 7.5$

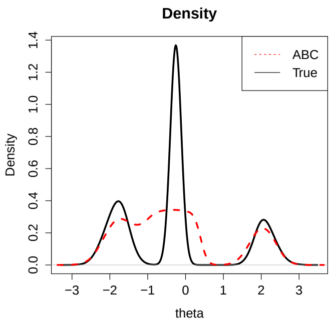

theta vs D
+ε
D
−ε
D
theta
20
10
0
−10
−3
−2
−1
0
1
2
3


Density
ABC
True
Density
theta
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
−3
−2
−1
0
1
2
3

$\epsilon = 5$

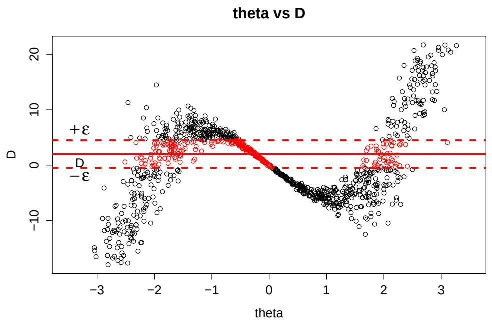


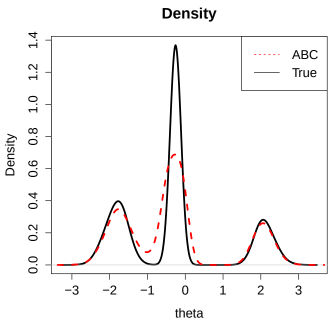
theta vs D
+ε
D
−ε
20
10
0
−10
−3
−2
−1
0
1
2
3
theta



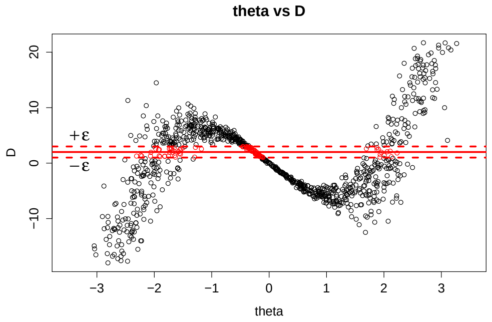
Density
ABC
True
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
−3
−2
−1
0
1
2
3
theta

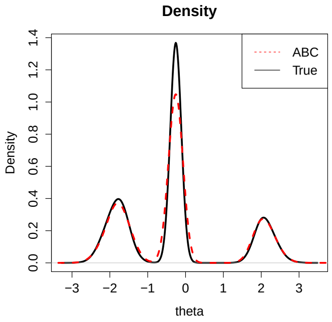
$\epsilon = 2.5$


theta vs D
D
theta
+ε
D
−ε
−3
−2
−1
0
1
2
3
−10
0
10
20


Density
Density
theta
ABC
True
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
−3
−2
−1
0
1
2
3

$\epsilon = 1$


theta vs D
D
20
10
0
−10
+ε
−ε
−3
−2
−1
0
1
2
3
theta


Density
Density
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
ABC
True
−3
−2
−1
0
1
2
3
theta

Summary statistics

If the data are too high dimensional we never observe simulations that are 'close' to the field data - curse of dimensionality

Reduce the dimension using summary statistics, $S(D)$.

Approximate Rejection Algorithm With Summaries

- Draw θ from $\pi(\theta)$
- Simulate $X \sim f(\theta)$
- Accept θ if $\rho(S(D), S(X)) < \epsilon$

If S is sufficient this is equivalent to the previous algorithm.

Summary statistics

If the data are too high dimensional we never observe simulations that are 'close' to the field data - curse of dimensionality

Reduce the dimension using summary statistics, $S(D)$.

Approximate Rejection Algorithm With Summaries

- Draw θ from $\pi(\theta)$
- Simulate $X \sim f(\theta)$
- Accept θ if $\rho(S(D), S(X)) < \epsilon$

If S is sufficient this is equivalent to the previous algorithm.

ABC is approximate for two reasons

- Using tolerance ϵ in $\rho(S(D), S(X)) < \epsilon$
- Using summary $S(D)$.

There is a trade-off:

- $\dim(S)$ small allows us to use small ϵ, but $\pi(\theta|s_{obs}) \not\approx \pi(\theta|D)$
- $\dim(S)$ large gives $\pi(\theta|s_{obs}) \approx \pi(\theta|D)$, but the ABC approximation is poor as curse of dimensionality forces us to use larger ϵ

Model selection

Consider comparing two models, $\mathcal{M}_1$ and $\mathcal{M}_2$.
Bayes factors (BF) are the Bayesian approach to model selection.

$$BF = \frac{\pi(D|\mathcal{M}_1)}{\pi(D|\mathcal{M}_2)}$$

where

$$\pi(D|\mathcal{M}_1) = \int \pi(D|\theta, \mathcal{M}_1)\pi(\theta)\mathrm{d}\theta$$

It is extremely challenging to calculate Bayes factors for even quite simple models.

- SMC^2, path-sampling, nested-sampling

Criterions such as the BIC are crude approximations to the BF.

Predictive evaluation using scoring rules looks to be a promising route.

Integrated nested Laplace approximation (INLA)

Rue, Martino, and Chopin, Ser. B, 2009

Computationally effective alternative to MCMC for Bayesian inference.
INLA is designed for latent Gaussian models, a wide and flexible class:

- regression models
- spatial and spatio-temporal models

$$
\begin{aligned}
\theta &\sim p(\theta) \\
x|\theta &\sim N(0, Q(\theta)^{-1}) \\
\eta &= c^\top x \\
y_i|x_i, \theta &\sim p(y_i|\eta_i, \theta)
\end{aligned}
$$

INLA will efficiently approximate $\pi(\theta|y)$ for low dimensional θ.

MCMC for Bayes Summary

- MCMC
  - most generally applicable gold standard method
- SMC/PF
  - primarily for time structured models or as an alternative to MCMC
- ABC
  - for models where all you can do is simulate (likelihood unknown)
- INLA
  - for latent Gaussian problems $(x|\theta)$ where you only care about marginal distributions $\pi(\theta|y)$

All of these methods require large number of simulator evaluations.

Resampling methods

Resampling methods

We often have a statistical procedure that we wish to evaluate.

- A parameter estimate - how confident are we in our estimate?
- A model which makes predictions - how accurate are the predictions?
- A hypothesis we wish to test - but don't know how.

There is no need for much of the classical statistical theory we teach - most of it was developed before computers and approximates what resampling methods do.

Bootstrapping

The bootstrap is a method for assessing properties of a statistical estimator in a *non-parametric* framework.

We use the data multiple times to generate 'new' data sets to assess the properties of parameters.

- Suppose we have data $X_1, \dots, X_n$ for which we want to estimate quantity $\theta(X)$
  - e.g. $\theta(X) = \mathbb{V}\text{ar}(X)$
- A bootstrap replicate dataset is generated by sampling from the data with replacement giving
  $$X_1^*, \dots, X_n^*$$
  and then calculating $\theta^* = \theta(X^*)$.

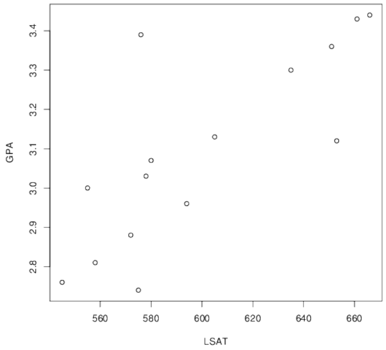
By repeating this a large number of times, giving $\theta_1^*, \theta_2^*, \dots$, we can assess the properties of $\theta(X)$

Lawschool example

A sample of 15 law schools was taken, and two measurements were made for each school:

x_i :	LSAT, average score for the class on a national law test
y_i :	GPA, average undergraduate grade-point average for the class

We are interested in the correlation coefficient between these two quantities, which we estimate to be $\theta = 0.776$.



How accurate is our estimate of the correlation coefficient?

Lawschool example - II

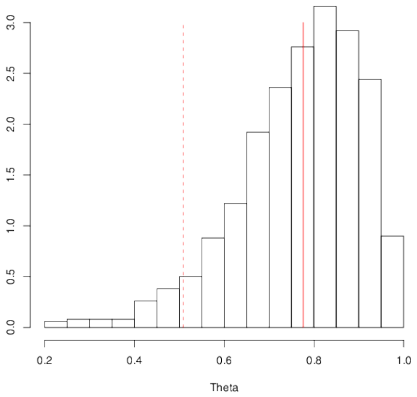
Use the bootstrap to estimate the standard error of $\theta = \mathbb{C}\text{or}(LSAT, GPA)$.

1. Sample 15 data points with replacement to obtain bootstrap data z^*.
2. Evaluate the sample correlation coefficient θ^* for the newly sampled data z^*.
3. Repeate steps 1 and 2 to obtain $\theta^{*(1)}, \ldots, \theta^{*(B)}$.
4. Estimate the standard error of the sample correlation coefficient by the sample standard deviation of $\theta^{*(1)}, \ldots, \theta^{*(B)}$.

Lawschool example - III

With $B = 1000$, we find the estimated standard error of θ to be 0.137.

- a histogram of the bootstrap replicates gives more information about the uncertainty about $\mathbb{C}\text{or}(LSAT, GPA)$.



Cross Validation

Cross validation is a useful computational tool for assessing the performance of a model in terms of its predictive ability.

This is generally in the context of regression or classification where we have trained the data using (x_i, y_i) pairs

Leave-one-out cross-validation For $i = 1, \ldots, n$

1. Fit the model to the reduced data set (or training set),
$$\{(x_1, y_1), \ldots, (x_{i-1}, y_{i-1}), (x_{i+1}, y_{i+1}), \ldots, (x_n, y_n)\}$$
2. Obtain from the fitted model the predicted value $\hat{y}_i$ at x_i.
3. Compute the squared error $\epsilon_i = (\hat{y}_i - y_i)^2$

The root mean square error can then be reported and used to compare models.

Monte Carlo and Permutation tests

Diet A	233	291	312	250	246	197	268	224
Diet B	185	263	246	224	212	188	250	148

Are the diets equally effective?

Monte Carlo and Permutation tests

Diet A	233	291	312	250	246	197	268	224
Diet B	185	263	246	224	212	188	250	148

Are the diets equally effective? A good test statistic might be

$$T = \bar{A} - \bar{B}$$

But we need the sampling distribution of T in order to do a hypothesis test.

Randomisation Test

1. Randomly re-assign the 16 individuals to the two groups.
2. Re-calculate the test-statistic for this permuted data
3. Repeat to obtain B sampled test-statistics $T_1, \ldots, T_B$.
4. For a two-sided test, the estimated p-value of the observed test statistic T_{obs} is

$$\frac{1}{B} \sum_{i=1}^{B} \mathbb{I}_{|T_i| \geq |T_{obs}|}$$

Using 10000 random permutations gave a p-value of 0.063.

Randomisation Test

1. Randomly re-assign the 16 individuals to the two groups.
2. Re-calculate the test-statistic for this permuted data
3. Repeat to obtain B sampled test-statistics $T_1, \ldots, T_B$.
4. For a two-sided test, the estimated p-value of the observed test statistic T_{obs} is

$$\frac{1}{B}\sum_{i=1}^{B}\mathbb{I}_{|T_i|\geq|T_{obs}|}$$

Using 10000 random permutations gave a p-value of 0.063.

The parametric test:

$$\text{Assume } X_i^{(j)} \sim N(\mu_j, \sigma^2)$$

The standard test is then a two sample t-test, based on the statistic

$$T = \frac{\bar{X}^{(1)} - \bar{X}^{(2)}}{\sqrt{s^2/8 + s^2/8}},$$

Under H_0, T has a t_{14}-distribution, giving a p-value of 0.0649.

Bayesian optimization

Black box (query only) model

$$x \longrightarrow f \longrightarrow y$$

Find $x^* = \arg\max f(x)$

Bayesian optimisation techniques use a surrogate model of $f(x)$ to do the optimisation.

- Used by Google, Facebook etc to fit their data models
- Basis of Deepmind and many machine learning methods.

Conclusions

- Computer power now allows Bayesian inference to be done for complex problems
- The calculations are not always cheap or simple
- Resampling methods allow us to implement frequentist procedures.

References

- Monte Carlo: Robert and Casella, Monte Carlo Statistical Methods, Springer, 2004
- MCMC: see above
- Particle methods: Doucet and Johansen 2010
- ABC: Marin, Pudlo, Robert, Ryder 2011
- INLA: Rue, Martino, and Chopin, Ser. B, 2009
- Resampling methods: Simon, Resampling: The new statistics, 1997
- Bayesian optimisation: Mockus, Bayesian approach to global optimisation: theory and applications, 2013