

Metropolis-Hastings

Metropolis-Hastings example.



Metropolis-Hastings: remarks

- Very simple and general.
- If the initial point is not close to the mass of $\pi(\theta | y)$, then we need to run the algorithm for a while, until we reach the mass of $\pi(\theta | y)$
 - "burn-in" or "warm-up" phase.
- Note that we never needed to calculate the marginal likelihood $p(y)$, since it cancels in the acceptance ratio.
- The algorithm is the same regardless of the dimension of θ .

Choosing the proposal

- A common proposal distribution is the random walk

$$q(\cdot \mid \theta) = \mathcal{N}(\cdot \mid \theta, \sigma_q^2),$$

where σ_q^2 is some variance.

- When σ_q^2 is very small, the proposed points will usually be accepted, but algorithm will only explore the space very slowly.
- When σ_q^2 is very large, we are proposing large changes to θ and can make large moves, but the proposed points will be rejected very often.
- The optimal σ_q^2 is not too small and not too large.
- However, as the dimension p of θ grows, the performance of this method becomes poor.

Gibbs sampler

- The Gibbs sampler is a special case of the Metropolis-Hastings algorithm.
- The idea is to avoid the problem of poor performance due to large p by changing only one variable at a time.
- Two ingredients:
 1. propose moves to one variable at a time
 2. use the full conditional distribution to make these proposals.

Full conditionals

- Suppose that we have a parameter vector $\theta = (\theta^1, \dots, \theta^p)$ and we wish to sample from the posterior $\pi(\theta | y)$.
- The full conditional for variable j is

$$\pi(\theta^j | \theta^{-j}, y).$$

- For some model specifications, the full conditional distribution may be of a known type, and so can be simulated from directly.

Full conditionals

- To see how this might be possible, assume prior independence of all θ^j

$$p(\theta) = p(\theta^1) \dots p(\theta^p).$$

- Then

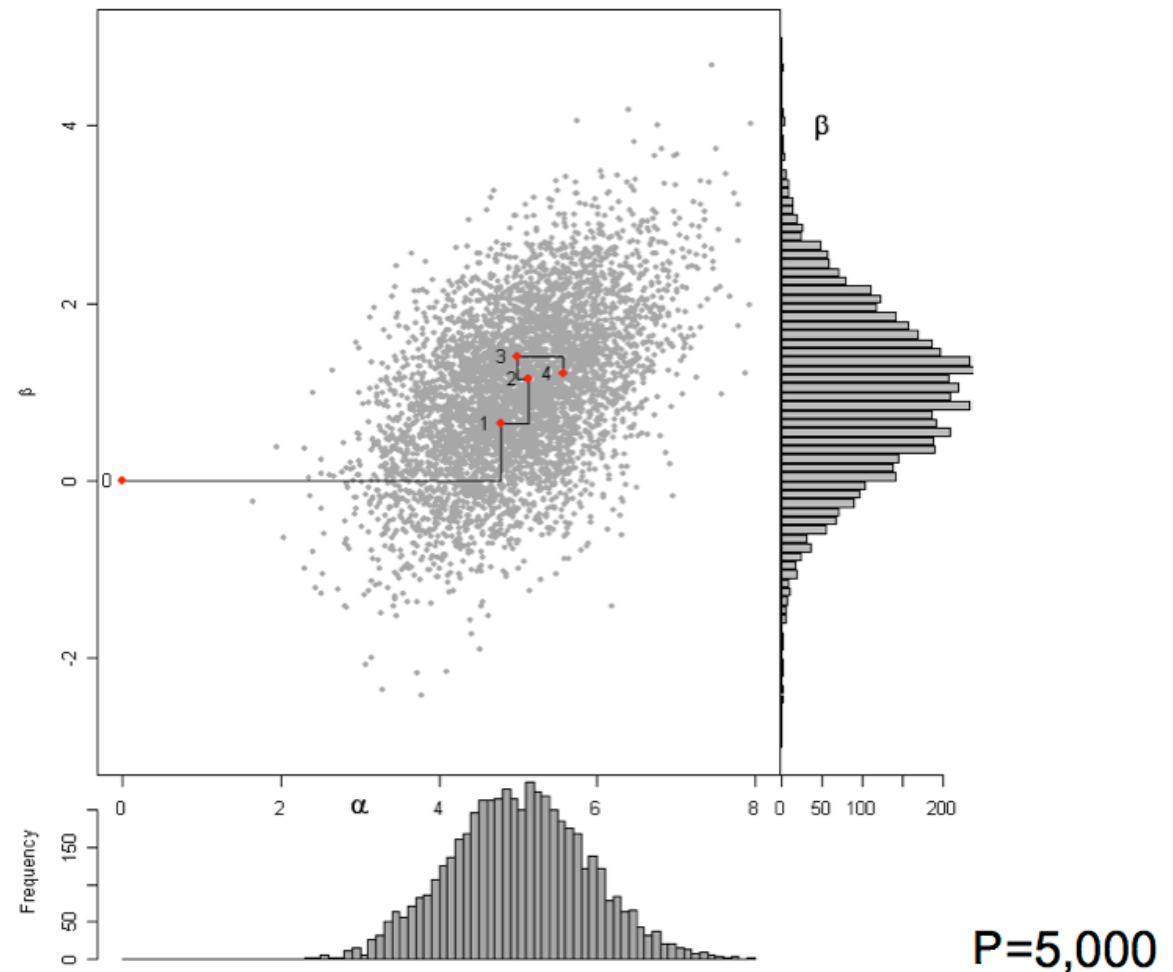
$$\begin{aligned}\pi(\theta^j | \theta^{-j}, y) &\propto p(\theta^1) \dots p(\theta^p) l(y | \theta) \\ &\propto p(\theta^j) l(y | \theta).\end{aligned}$$

- If $p(\theta^j)$ is conjugate to $l(y | \theta)$ with θ^{-j} fixed, then $\pi(\theta^j | \theta^{-j}, y)$ has the same form as the prior.

Gibbs sampler

- Simply draw from each full conditional in turn.
- This is a special case of Metropolis-Hastings where
 - the proposals used are the full conditional distributions
 - this gives an acceptance rate of 1 for every move.
- This method has been very successful and widely used
 - see the BUGS and JAGS packages.
- We will see a concrete example of this when we cover Bayesian variable selection.

Gibbs sampler example



The weakness of the Gibbs sampler

- When the θ variables are strongly dependent, the Gibbs sampler explores the space very slowly.
- This is a fatal flaw for some models:
 - for example, a regression model where the predictors are sites in genetic data (which can be strongly dependent), and where p is large.

Summary

- We have introduced the most commonly used MCMC methods.
- Next we will look at Bayesian variable selection
 - we will use these techniques for inference in these settings
 - compare with the frequentist approach for tackling the same problem
 - we will see an example of why doing Bayesian inference when p is large is such a challenging problem.

Bayesian linear regression

ST420 Lecture 15

Richard Everitt

Recap and motivation

- We have introduced Bayesian statistics, and the most important technique for Bayesian computation, Monte Carlo methods.
- We have seen that Bayesian methods may be used for regularisation
 - ridge is a MAP estimator with a Gaussian prior
 - lasso is a MAP estimator with a Laplace prior.
- When looking at large p problems, we discussed the importance of variable selection.
- How do we do Bayesian linear regression?
- Can we do Bayesian variable selection?

This lecture

- Introduce Bayesian linear regression.
- Review best subset selection.
- Review Bayesian model comparison.
- Describe a model for Bayesian variable selection.
- Review Gibbs samplers.
- Describe a Gibbs sampler for Bayesian variable selection.

Bayesian linear regression

- Suppose we model the data using the linear regression $y = x^T \beta + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$.
- We can use least squares (or equivalently maximum likelihood) to estimate the parameters.
- What is Bayesian linear regression?
 - treat β and σ^2 as random variables, and find their posterior distribution $\pi(\beta, \sigma^2 | y)$.
- Why do this instead of the frequentist method?
 - we can incorporate any prior knowledge we have into the analysis
 - we get an accurate reflection of how certain we are about the fit
 - predictions (or other use of the model) can account for the uncertainty we have about the model fit.
- Note that this also has advantages over MAP
 - we are now inferring the whole posterior, not just the mode.

What do we have to do?

- Choose priors p . (We will see that the model l is specified by the fact we are using linear regression with Gaussian noise.)
- Work out how to get the posterior $\pi(\beta, \sigma^2 \mid y) \dots$
 - we have multiple parameters, so we are unlikely to have a fully conjugate prior
 - this means we are going to need to use Monte Carlo (sampling)
 - (actually, this is one of the very few situations with multiple parameters where we can find the posterior analytically, but we disregard this for expository purposes).
- We will use a Gibbs sampler.

What do we have to do? (continued)

- How do we do construct the sampler?
 - we need to find the full conditional distribution of each variable
 - if a full conditional is a known distribution type, we can sample from it directly (Gibbs sampling)
 - (if it is not, then we need to propose the variable from some proposal, and perform an accept-reject step "Metropolis-within-Gibbs").
- I've chosen my priors, constructed my sampler, coded it up, and run it. Am I done?
- Not quite...We need to check that our MCMC has converged to the posterior, and that we have run it for long enough after convergence
 - more about this later.

Model and priors

- What is the distribution of $y \mid \beta, \sigma^2, x$?
 - we omit x from the conditioning from hereon, since it is fixed.
- Using standard properties of a normal distribution, we have $y \mid \beta, \sigma^2 \sim \mathcal{N}(x^T \beta, \sigma^2)$.
- The unknowns are β and σ^2 . The maths is slightly simpler if we use the parameterisation $\tau = 1/\sigma^2$ (where τ is known as the “precision”).
- A standard choice of priors is $\beta \sim \mathcal{N}(0, S)$ (where S is a covariance matrix) and $\tau \sim \Gamma(a, b)$.

Posterior

- We need the posterior

$$\begin{aligned}\pi(\beta, \tau \mid y) &\propto p(\beta, \tau, y) \\ &= p(\beta)p(\tau)l(y \mid \beta, \tau) \\ &= p(\beta)p(\tau) \prod_{i=1}^n l_i(y_i \mid \beta, \tau).\end{aligned}$$

- First, find the full conditionals of β and τ (will be on the problem sheet).

Full conditionals

- For τ ,

$$\pi(\tau \mid y, \beta) \propto p(\tau) \prod_{i=1}^n l_i(y_i \mid \beta, \tau),$$

which gives

$$\tau \mid y, \beta \sim \Gamma\left(a + n/2, b + \sum_{i=1}^n \frac{(y_i - x_i^T \beta)^2}{2}\right).$$

Full conditionals

- For β ,

$$\pi(\beta \mid y, \tau) \propto p(\beta) \prod_{i=1}^n l_i(y_i \mid \beta, \tau),$$

which gives

$$\beta \mid y, \tau \sim \mathcal{N}\left(\left(S^{-1} + \tau \mathbf{X}^T \mathbf{X}\right)^{-1} \tau \mathbf{X}^T y, \left(S^{-1} + \tau \mathbf{X}^T \mathbf{X}\right)^{-1}\right),$$

where \mathbf{X} is the usual design matrix.

Gibbs sampler

- We can simulate directly from each of the full conditionals, so we may use a Gibbs sampler.
- Let $(\beta_{(0)}, \tau_{(0)})$ be an initial value of the chain.
- For each $1 < t \leq N$, where N is some maximum number of iterations:
 - simulate $\tau_{(t)}$ from its full conditional $\Gamma\left(a + n/2, b + \sum_{i=1}^n \frac{(y_i - x_i^T \beta_{(t-1)})^2}{2}\right)$
 - simulate $\beta_{(t)}$ from its full conditional $\mathcal{N}\left(\left(S^{-1} + \tau_{(t)} \mathbf{X}^T \mathbf{X}\right)^{-1} \tau_{(t)} \mathbf{X}^T y, \left(S^{-1} + \tau_{(t)} \mathbf{X}^T \mathbf{X}\right)^{-1}\right)$.
- Discard the points $(\beta_{(t)}, \tau_{(t)})$ where $t < B$ for B being some “burn in”.

Best subset selection

- Rather than shrinking the β values, make a "hard" decision about which of the p variables to include.
- Known as *Best Subset Selection*.
- Usually implemented using a variation on the following idea
 - find the best model (using ERM) when using only one predictor (i.e. try using each predictor on its own, and choose the one that minimises the empirical risk)
 - do the same for all models with 2 predictors, then 3, etc
 - use some approach that penalises model complexity (e.g. AIC, etc) to choose the model complexity.
- Problem: there are 2^p models
 - often cannot search through exhaustively
 - search algorithms are usually greedy and ad hoc.

Bayesian variable selection

- In Bayesian statistics, everything that is unknown is treated as a random variable
 - this includes which variables to accept.
- We have 2^p models
 - denote these by M^1, M^2, \dots, M^{2^p} .
- We could make a discrete variable M whose possible values are these models
 - then find $\pi(M | y)$.

Bayesian variable selection

- Complication
 - these models all have different parameters, about which we are also uncertain
 - e.g. M^i could be

$$y \sim \mathcal{N}(\beta^0 + \beta^5 x^5 + \beta^{52} x^{52} + \beta^{53} x^{53}, \sigma^2)$$

- M^j could be

$$y \sim \mathcal{N}(\beta^0 + \beta^{53} x^{53}, \sigma^2).$$

Bayesian model comparison

- Bayesian statistics has a built-in method for model comparison.
- Simply treat the model M as a random variable, with prior distribution $p(M)$.
- The relative posterior probability of two models M^1 and M^2 is known as the *posterior odds* and is given by

$$\frac{\pi(M^1 | y)}{\pi(M^2 | y)} = \underbrace{\frac{p(M^1)}{p(M^2)}}_{\text{prior odds}} \underbrace{\frac{l(y | M^1)}{l(y | M^2)}}_{\text{Bayes factor}}.$$

- Where does $l(y | M)$ come from?
 - this is the evidence for model M

$$l(y | M) = \int_{\theta_M} p(\theta_M) l(y | \theta_M) d\theta_M.$$

Bayesian model comparison for variable selection

- Using Bayes' theorem

$$\pi(M \mid y) \propto p(M)l(y \mid M).$$

- This gives us a score for each model
 - often the prior might be chosen to uniform
 - so the important object for scoring each model is the evidence $l(y \mid M)$
 - M is a discrete set, so we can easily calculate the normalising constant in Bayes' theorem.
- "All" we need to do is to calculate $l(y \mid M)$ for each of the 2^p models.
- However:
 - $l(y \mid M)$ could be an intractable integral;
 - 2^p models could be rather too many in practice.

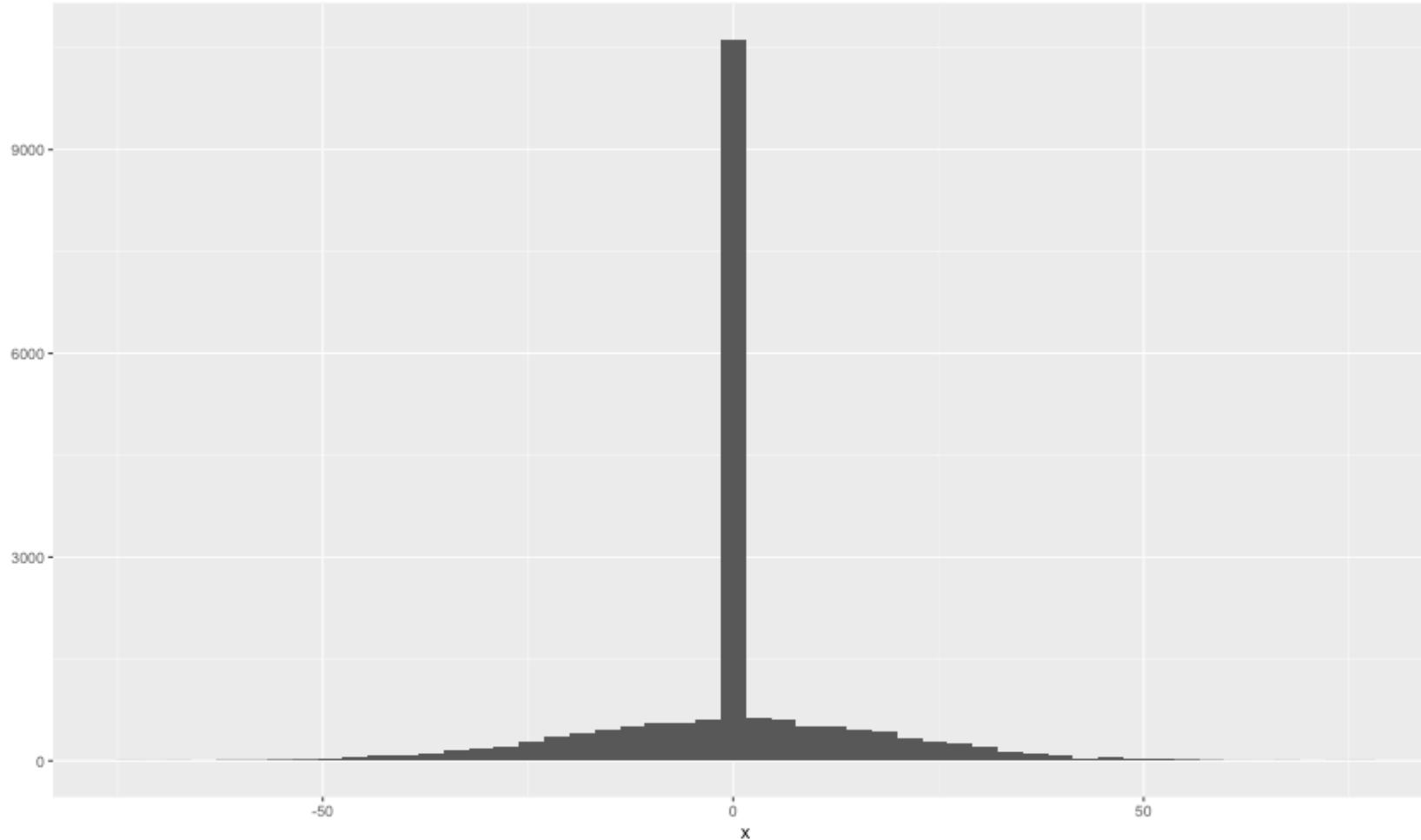
Bayesian variable selection

- Although $l(y | M)$ is intractable in general, we will see that for linear models we can find it analytically.
- Rather than visiting all 2^p models, we will sample from $\pi(M | y)$
 - we hope that through sampling, we can spend our computational budget on exploring the models that are possible explanations of the data, rather than those that have close to zero posterior probability.

Bayesian model

- Rather than enumerating every model, we use a different parameterisation.
- Suppose, as usual, we have n data points, p predictors and one intercept term, and use $y \sim \mathcal{N}(x^T \beta, \sigma^2)$.
- We define a prior on the β variables that
 - gives some probability to them being zero
 - some probability to them being non-zero
 - except for the intercept term that we assume is always in the model.
- This is sometimes called a "spike-and-slab" model
 - the spike is a Dirac delta distribution at 0
 - the slab is usually a normal distribution centred at zero
 - there needs to be some weight that defines the relative contributions of the spike and the slab.

Draws from a spike-and-slab prior



Bayesian model

- The spike-and-slab prior is usually parameterised in the following way.
- Let γ be a vector of p binary variables $\gamma = (\gamma^1, \dots, \gamma^p)$
 - $\gamma^j = 1$ will represent the presence of $\beta^j x^j$ in the model, and $\gamma^j = 0$ will represent its absence.

Bayesian model

- Define a prior distribution $p(\gamma)$
 - a common choice is

$$p(\gamma) = \prod_{j=1}^p p(\gamma^j),$$

independent prior distributions on the inclusion of each predictor

- for each γ^j , $p(\gamma^j = 1) = \pi^j$ for some $0 \leq \pi^j \leq 1$
- a common choice is to choose $\pi^j = \pi$, the same for all j
- the choice of π will have an influence on the number of variables we will end up seeing in the model.

Bayesian model

- We choose σ^2 , the variance of $y \mid x$, to be fixed.
- Given γ , we set β^γ to be the subvector of β for which the corresponding values of γ are equal to 1.
- We use a Gaussian prior on β^γ , i.e. $p(\beta^\gamma \mid \gamma) = \mathcal{N}(\mu^\gamma, S^\gamma)$
 - mean $\mu^\gamma = \mathbf{0}^\gamma$, a vector of zeros that has length of the number of γ equal to 1
 - and covariance $S^\gamma = \sigma^2 \mathbf{I}^\gamma$, a constant σ^2 times the identity matrix with the dimension being the number of γ equal to 1.

Inference (fitting)

- The space of models is parameterised by γ .
- We wish to sample from

$$\pi(\gamma \mid y) \propto p(\gamma) l(y \mid \gamma).$$

- In order to do this sampling, we need to find $l(y \mid \gamma)$.
- The output of our sampling will be realisations of the vector γ : $\gamma_1, \gamma_2, \dots, \gamma_N$
 - this simply gives the posterior distributions of which parameters will be included in the model.
- Once we have this sample, we will see that it is easy to obtain the posterior distribution on the parameters for each model, i.e. $\pi(\beta^{\gamma_i} \mid \gamma_i, y)$ for sample i , γ_i from $\pi(\gamma \mid y)$.

Posterior of β^γ

- It is easy to examine the latter problem first.
- What is $\pi(\beta^\gamma \mid \gamma, y)$?
- We know that

$$\pi(\beta^\gamma \mid \gamma, y) \propto p^\gamma(\beta^\gamma \mid \gamma) l^\gamma(y \mid \beta^\gamma, \gamma),$$

where

$$l^\gamma(y \mid \beta^\gamma, \gamma) = \mathcal{N}\left(y \mid (x^\gamma)^T \beta^\gamma, \sigma^2\right)$$

and

$$p^\gamma(\beta^\gamma \mid \gamma) = \mathcal{N}(\mathbf{0}^\gamma, s^2 \mathbf{1}^\gamma).$$

Conditional posterior of $\beta_{\gamma_i} \mid \gamma$

- A normal prior on β^γ is a conjugate prior in this situation.
- Therefore $\pi(\beta^\gamma \mid \gamma, y)$ is a normal distribution.
- We obtain

$$\pi(\beta^\gamma \mid \gamma, y) = \mathcal{N}\left(\left(\frac{1}{s^2} \mathbf{1}^\gamma + \frac{1}{\sigma^2} (\mathbf{X}^\gamma)^T \mathbf{X}^\gamma\right)^{-1} \frac{1}{\sigma^2} (\mathbf{X}^\gamma)^T y, \left(\frac{1}{s^2} \mathbf{1}^\gamma + \frac{1}{\sigma^2} (\mathbf{X}^\gamma)^T \mathbf{X}^\gamma\right)^{-1}\right)$$

(a similar derivation will be on the problem sheet).

- Conditional on γ , the posterior on β^γ is simply a normal distribution.

Computing $l(y \mid \gamma)$

- $l(y \mid \gamma)$ is the marginal likelihood for the model γ .
- We have

$$l(y \mid \gamma) = \int_{\beta^\gamma} p^\gamma(\beta^\gamma \mid \gamma) l^\gamma(y \mid \beta^\gamma, \gamma) d\beta^\gamma.$$

- One method for computing this is to plug in the expressions for p^γ and l^γ , then to manipulate the expressions so that one can see that the integrand is a Gaussian density, which integrates to 1.
- Another is to use

$$l(y \mid \gamma) = \frac{p^\gamma(\beta^\gamma \mid \gamma) l^\gamma(y \mid \beta^\gamma, \gamma)}{\pi(\beta^\gamma \mid \gamma, y)}.$$

Computing $l(y \mid \gamma)$

- $l(y \mid \gamma)$ is given by

$$\frac{\mathcal{N}(y \mid (x^\gamma)^T \beta^\gamma, \sigma^2) \mathcal{N}(\mathbf{0}^\gamma, s^2 \mathbf{1}^\gamma)}{\mathcal{N}\left(\left(\frac{1}{s^2} \mathbf{1}^\gamma + \frac{1}{\sigma^2} (\mathbf{X}^\gamma)^T \mathbf{X}^\gamma\right)^{-1} \frac{1}{\sigma^2} (\mathbf{X}^\gamma)^T y, \left(\frac{1}{s^2} \mathbf{1}^\gamma + \frac{1}{\sigma^2} (\mathbf{X}^\gamma)^T \mathbf{X}^\gamma\right)^{-1}\right)}.$$

- Simplifying this is messy, and doesn't help us understand very much.
- The key point is that we have an analytic expression for $l(y \mid \gamma)$
 - this allows us to set up an MCMC that simulates from $\pi(\gamma \mid y)$.

Summary

- We have introduced Bayesian linear regression.
- We have done a lot of the work in developing Bayesian variable selection
 - but we have not yet described the sampler we will use.
- Next time we will
 - discuss the MCMC schemes used for Bayesian variable selection
 - also examine other Bayesian regularisation schemes and the MCMC algorithms used to fit them.

Bayesian variable selection and regularisation

ST420 Lecture 16

Richard Everitt

Recap

- We have introduced Bayesian approaches to regression
 - similar ideas may be used for classification.
- Our theme has been using Bayes for variable selection and regularisation
 - we introduced a model for Bayesian variable selection
 - we started to work towards MCMC methods for fitting this model.
- We are working towards a topic at the forefront of current research
 - MCMC in high dimensions
 - will be covered towards the end of the module.

Variable selection model

- Use a spike-and-slab model for β .
- Parameterised by a random vector γ that indicates the presence or absence of each variable.
- Spike-and-slab uses a Gaussian prior on the β that are present
 - means that we can find $l(y \mid \gamma)$ analytically.
- We would like to set up a sampler on $\pi(\gamma \mid y)$ that explores the 2^p states of the γ vector.

Aside: extensions to the model

- We can treat σ^2 as unknown, and use a Gamma prior.
- If we are uncertain as to how to choose π , we can treat it as unknown, and set a uniform prior $p(\pi)$ on $[0, 1]$.
- We can use a different mean and variance in the Gaussian prior on β
 - an "empirical Bayes" choice is popular, where the data is used to help set the prior, for example setting the mean of the Gaussian to be the least squares estimate.
- In each of these cases $l(y | \gamma)$ can still be found analytically.

Sampling from $\pi(\gamma \mid y)$

- How can we sample from $\pi(\gamma \mid y)$?
- γ is discrete with 2^p states
 - we could simply calculate the probability of all of the states, and sample from this discrete distribution
 - not computationally feasible.

Sampling from $\pi(\gamma \mid y)$

- Can we use Metropolis-Hastings to sample from $\pi(\gamma \mid y)$?
 - we would make a proposal γ^* , then calculate $p(\gamma^*)l(y \mid \gamma^*)$ when finding the acceptance ratio
 - cost is $O(p)$ per iteration.
- How many iterations might we need?
 - we need fewer iterations if the proposal is "good"
 - the proposal is "good" if it proposes γ in regions where $\pi(\gamma \mid y)$ has more mass.

Metropolis-Hastings on $\pi(\gamma \mid y)$

- We need a proposal distribution $q(\gamma^* \mid \gamma)$ to propose candidate values.
- Suppose that there is no dependence between any of the γ^j in the posterior
 - this is not something we would know until we have done the sampling
 - this would mean that

$$\pi(\gamma \mid y) = \prod_{j=1}^p \pi(\gamma^j \mid y).$$

- If the current value of γ in our Metropolis-Hastings has a high probability, we see that for γ^* to be accepted
 - the proposed value for γ^1 needs to be such that $\pi(\gamma^1 \mid y)$ is not close to 0
 - the proposed value for γ^2 needs to be such that $\pi(\gamma^2 \mid y)$ is not close to 0
 - and so on.

Metropolis-Hastings on $\pi(\gamma | y)$

- We see that that probability of rejecting is determined by the probability that we have at least one "failure" when proposing the γ^j .
- If the probability of success for each γ^j is ρ , we have that the probability of at least one failure is

$$1 - \rho^p.$$

- this probability very quickly grows to 1 as p grows.
- Thus we see that for this problem, Metropolis-Hastings is doomed to failure as p gets large.
- The problem is that we are proposing all values of γ simultaneously
 - therefore we could try using a Gibbs sampler which proposes one at a time.

Gibbs sampler on $\pi(\gamma \mid y)$

- Suppose we use a Gibbs sampler, which moves each of the γ^j one at a time.
- To move γ^j , we need to draw from

$$\pi(\gamma^j \mid \gamma^{-j}, y).$$

- We use

$$\pi(\gamma^j \mid \gamma^{-j}, y) \propto p(\gamma) l(y \mid \gamma).$$

Gibbs sampler on $\pi(\gamma \mid y)$

- Note that there are only two possible values for γ^j . So, we simply need to evaluate $p(\gamma)l(y \mid \gamma)$ where
 - the γ^{-j} are fixed to be their current values from the chain
 - we consider the two possibilities, that $\gamma^j = 1$ and that $\gamma^j = 0$.
- Looking at $p(\gamma)l(y \mid \gamma)$ in these two cases will give us an unnormalised binary distribution
 - at each step of the Gibbs sampler, we simply normalise this distribution, then sample our new values of γ^j from it.

Gibbs sampler on $\pi(\gamma \mid y)$

- How well will the Gibbs sampler work?
- Its performance will be reasonable, unless the γ are highly dependent in the posterior
 - when only the joint proposal to include some subset of variables would achieve a large value of $p(\gamma)l(y \mid \gamma)$
 - i.e. if we proposed the individual inclusion of variables in this subset, we would not achieve the same large value of $p(\gamma)l(y \mid \gamma)$.
- This may be the case in the genetics example we saw earlier in the module.

Samplers for variable selection

- The performance of Metropolis-Hastings degrades when the dimension of the space on which we perform the proposals grows.
- The performance of Gibbs sampling will be poor when the variables have a strong dependence in the posterior
 - unfortunately we do not know if this is the case until we use the sampler!
- Ideally we might jointly update blocks of dependent variables using Metropolis-Hastings
 - but how do we know which variables to block together?

Alternative priors for regularisation

- There are ideas (e.g. "adaptive" MCMC) that can help with this problem, but sampling from a discrete space of size 2^p remains challenging.
- It can be a bit easier to construct samplers to sample from continuous spaces when p is large
 - see later in the module.
- Remaining idea for today:
 - examine other priors that result in regularisation.

Some ideas

- We have already mentioned Gaussian or Laplace priors on β , but there are many other options, e.g. Student-t, etc.
- One commonly used idea is to use a "global-local" construction:
 - use a hierarchical model of the form

$$\beta^j \sim \mathcal{N}\left(0, (\lambda^j)^2 \sigma_\beta^2\right),$$

$$(\lambda^j)^2 \sim p((\lambda^j)^2),$$

$$(\sigma_\beta^2, \sigma^2) \sim p(\sigma_\beta^2, \sigma^2).$$

- σ_β^2 provides "global" shrinkage across all coefficients.
- $(\lambda^j)^2$ provides variation in the priors for different β^j .

Horseshoe prior

- In this prior, a half Cauchy prior is chosen on λ^j

$$\lambda^j \sim C^+(0, 1).$$

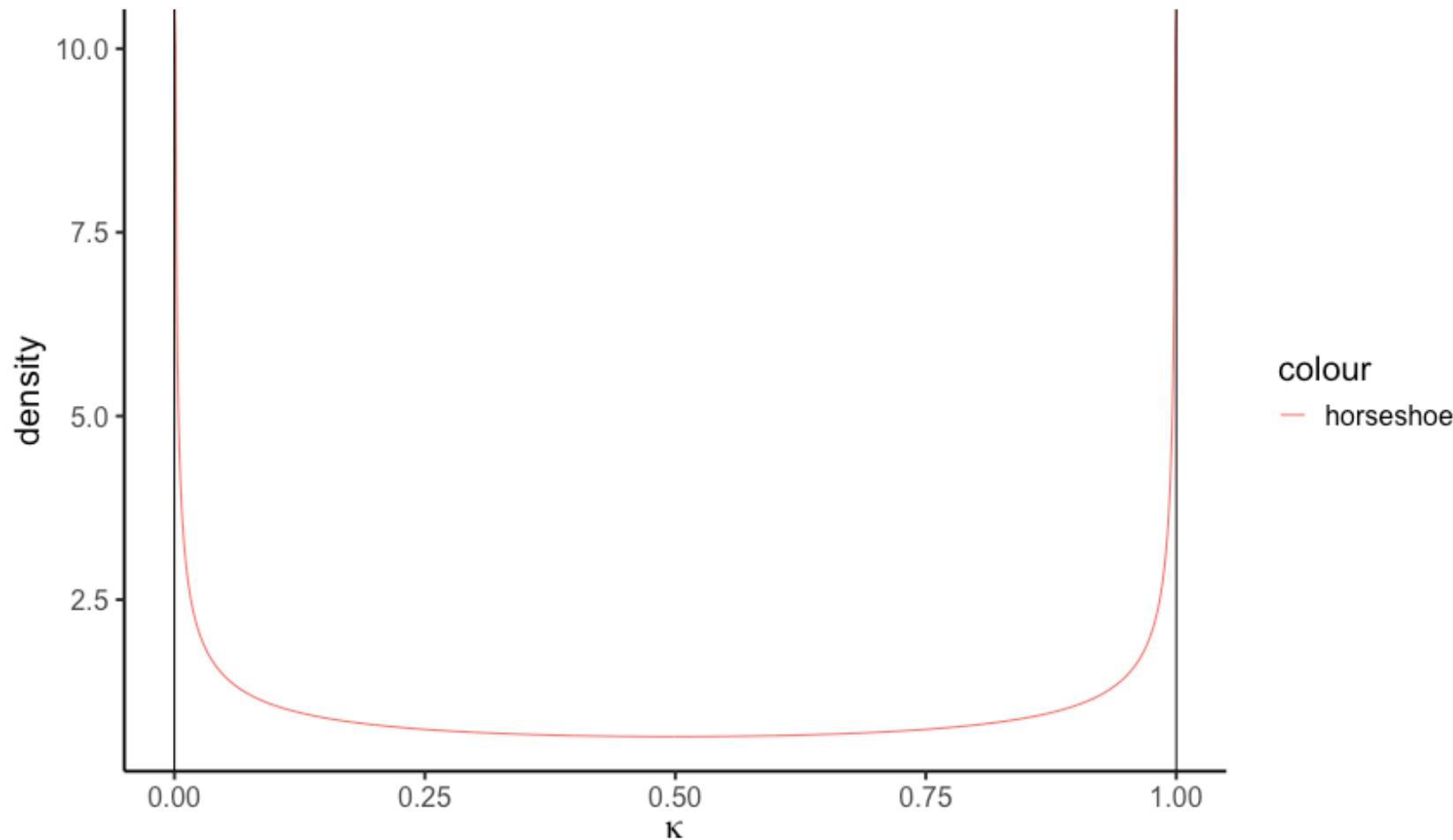
- A Cauchy distribution has heavy tails
 - the idea is that whilst the global shrinkage shrinks all of the coefficients towards zero the heavy tailed Cauchy will allow some of the coefficients to "escape" the shrinkage.
- Approximately we have, under the global-local priors, that the posterior mean $\bar{\beta}^j$ of each β^j satisfies

$$\bar{\beta}^j = (1 - \kappa^j) \hat{\beta}^j \quad \kappa^j = \frac{1}{1 + n\sigma^{-2}(\lambda^j)^2},$$

where $\hat{\beta}^j$ is the maximum likelihood estimator of β^j .

- For the horseshoe, the induced prior on κ^j is $\text{Beta}(0.5, 0.5)$.

Horseshoe prior



Horseshoe prior

- The vertical black lines on the previous plot represent, for comparison, the prior we used in variable selection
 - although it is not strictly performing variable selection, the horseshoe can be seen as a continuous distribution that accomplishes a similar effect.
- Advantage:
 - for inference (fitting) we do not need to design a sampler that explores a discrete space
 - we can use ideas that use the gradient of the target distribution, such as the Langevin algorithm or Hamiltonian Monte Carlo (towards the end of the module).

PAC bounds

ST420 Lecture 17

Richard Everitt

Recap

- We began by introducing simple methods for regression and classification.
- We then formalised several ideas
 - loss/risk/empirical risk for fitting models and for quantifying prediction error
 - excess risk → estimation error → generalisation error.
- We introduced a number of the main techniques for classification.
- Then we had an aside, looking at Bayesian methods.

Plan for the next few weeks

- Further develop the theory underpinning supervised learning
 - find bounds on the excess risk, estimation error and generalisation error
 - discuss some of the practical implications of these bounds.
- Introduce resampling approaches
 - cross validation
 - bootstrap
 - bagging
 - boosting.

Learning theory: the story so far

- X and Y are random variables, and P is their joint distribution (which is unknown).
- $f \in \mathcal{F}$ is a function from \mathcal{X} to \mathcal{Y} .
- Loss:

$$L : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0}$$

gives a distance $L(Y, f(X))$ between the real Y and predictions $f(X)$.

- Risk:

$$R(f) = \mathbb{E}_{X,Y} [L(Y, f(X))]$$

is the expected loss over P .

- Denote the optimal $f \in \mathcal{F}$, i.e. the function achieves the minimum risk according to the chosen loss, by f^* .

Learning theory: the story so far

- Training data \mathcal{T} is a random sample $\{(X_i, Y_i)\}_{i=1}^n$
- Empirical risk: estimate of risk based on training data

$$\hat{R}(f) = \frac{1}{n} \sum_{i=1}^n L(Y_i, f(X_i)).$$

- Consider a restricted space of functions $\bar{\mathcal{F}}$.
- Choose the f that minimises the empirical risk: call it \hat{f} .

Learning theory: the story so far

- Study the *excess risk*

$$R(\hat{f}) - R(f^*).$$

- Decomposing excess risk:

$$R(\hat{f}) - R(f^*) = \underbrace{\left(R(\hat{f}) - R(\bar{f}^*) \right)}_{\text{estimation error}} + \underbrace{\left(R(\bar{f}^*) - R(f^*) \right)}_{\text{approximation error}}.$$

- *Estimation error* compares the risk of the best possible f - call it \bar{f}^* - in our choice of function space to the risk of our estimate obtained using ERM.
- *Approximation error* compares the risk of the best possible f to the best f in our choice of function space.
- Focus on estimation error.

Learning theory: the story so far

- Estimation error can be decomposed as follows

$$R(\hat{f}) - R(\bar{f}^*) = \underbrace{\left(R(\hat{f}) - \hat{R}(\hat{f}) \right)}_{(1)} + \underbrace{\left(\hat{R}(\hat{f}) - \hat{R}(\bar{f}^*) \right)}_{(2)} + \underbrace{\left(\hat{R}(\bar{f}^*) - R(\bar{f}^*) \right)}_{(3)}.$$

- We know that $\hat{R}(\hat{f}) \leq \hat{R}(\bar{f}^*)$ by definition of \hat{f} , so (2) ≤ 0 .
- Thus

$$R(\hat{f}) - R(\bar{f}^*) \leq \underbrace{\left(R(\hat{f}) - \hat{R}(\hat{f}) \right)}_{(1)} + \underbrace{\left(\hat{R}(\bar{f}^*) - R(\bar{f}^*) \right)}_{(3)}.$$

Learning theory: the story so far

- Term (3) is $\hat{R}(\bar{f}^*) - R(\bar{f}^*)$.
- Term (1) is $R(\hat{f}) - \hat{R}(\hat{f})$: the *generalisation error* of \hat{f} .
- (3) and (1) have the same form.
- We focus on bounding the size of the generalisation error $R(f) - \hat{R}(f)$ of a function f
 - this covers both (1) and (3).

This lecture

- We put generalisation error to one side for this lecture.
- In this lecture we bound excess risk, $R(\hat{f}) - R(f^*)$, in a special case
 - binary classification with $\mathcal{Y} = \{0, 1\}$
 - zero-one loss.
- Assume $R(\bar{f}^*) = 0$
 - this is a strong assumption!
 - we are assuming that our restricted set of functions $\bar{\mathcal{F}}$ contains the optimal f , i.e. $\bar{f}^* = f^*$
 - also that the "Bayes risk" $R(f^*)$ is zero - we can find a perfect classifier.
- Assume $\bar{\mathcal{F}}$ is finite.
- We will look at a *probably approximately correct* (PAC) bound.

Our first PAC bound

- We use the same notation as throughout the course, so \hat{f} is the function obtained through ERM with n training points.
- **Theorem (Valiant, 1984):** For all $n > 0$ and for all $\varepsilon > 0$

$$\mathbb{P}\left(R\left(\hat{f}\right) > \varepsilon\right) < |\bar{\mathcal{F}}|e^{-n\varepsilon} \equiv \delta.$$

- When $R\left(\bar{f}^*\right) = 0$, the excess risk is simply $R\left(\hat{f}\right)$.
- The probability is less than δ ("probably") that the excess risk is greater than ε ("approximately correct").

Aside: union bound

- The *union bound*, also known as *Boole's inequality* says that for a countable set of events A_1, A_2, \dots ,

$$\mathbb{P}\left(\bigcup_i A_i\right) \leq \sum_i \mathbb{P}(A_i).$$

- The union bound can be generalised to give upper and lower bounds on the probability of finite unions of events
 - known as *Bonferroni inequalities*.

Proof of Valiant 1984

- For zero-one loss, $R(f) = P(f(X) \neq Y)$. Therefore if $R(f) > \varepsilon$, $P(f(X) \neq Y) > \varepsilon$ and hence $P(f(X) = Y) < 1 - \varepsilon$.
- For $\hat{R}(f) = 0$, we need that all of the classifications of the training points are perfect, i.e. $\forall i, f(X_i) = Y_i$. Therefore, using the above, if $R(f) > \varepsilon$, $\mathbb{P}(\hat{R}(f) = 0) < (1 - \varepsilon)^n$.
- Recall that $R(f) = \mathbb{E}[\hat{R}(f)]$. For any f , $\hat{R}(f) \geq 0$. Therefore for any f where $R(f) = 0$, we have $\hat{R}(f) = 0$.
- \hat{f} minimises \hat{R} , and the minimum value is 0, so we have $\hat{R}(\hat{f}) = 0$. Note that the minimum of \hat{R} is not necessarily unique: let $\hat{\mathcal{F}} = \left\{ f \in \bar{\mathcal{F}} \mid \hat{R}(f) = 0 \right\}$. We simply choose \hat{f} to be one of the members of $\hat{\mathcal{F}}$.
- Then...

Proof of Valiant 1984

$$\begin{aligned}\mathbb{P}\left(R\left(\hat{f}\right) > \varepsilon\right) &\leq \mathbb{P}\left(\bigcup_{f \in \hat{\mathcal{F}}} \{R(f) > \varepsilon\}\right) \\ &= \mathbb{P}\left(\bigcup_{f \in \bar{\mathcal{F}}} \left\{ R(f) > \varepsilon \text{ and } \hat{R}(f) = 0 \right\}\right) \\ &= \mathbb{P}\left(\bigcup_{f \in \bar{\mathcal{F}} | R(f) > \varepsilon} \left\{ \hat{R}(f) = 0 \right\}\right) \\ &\leq \sum_{f \in \bar{\mathcal{F}} | R(f) > \varepsilon} \mathbb{P}\left(\hat{R}(f) = 0\right) < \sum_{f \in \bar{\mathcal{F}} | R(f) > \varepsilon} (1 - \varepsilon)^n \\ &\leq |\bar{\mathcal{F}}| \cdot (1 - \varepsilon)^n \leq |\bar{\mathcal{F}}| e^{-n\varepsilon}\end{aligned}$$

Meaning of the bound

- The bound depends on the size $|\bar{\mathcal{F}}|$ of the function class, and the size n of the training data.
- We can think of $|\bar{\mathcal{F}}|$ as being the complexity/flexibility of the function class
 - if there are more models, the class is more flexible.
- We can think of ε as being the *accuracy* we are guaranteed, since it is a bound on the excess risk.
- We can think of δ as being the *confidence* we have in the accuracy, since it bounds the probability that we are not accurate.
- For large n , $\delta = |\bar{\mathcal{F}}|e^{-n\varepsilon}$ can be made arbitrarily small
 - as we get more training data, we can be arbitrarily confident in our accuracy.

Rewriting the bound

- If we choose

$$n \geq \frac{\log |\bar{\mathcal{F}}| + \log(1/\delta)}{\varepsilon}$$

then with probability at least $1 - \delta$, $R(\hat{f}) \leq \varepsilon$ (proof: exercise).

- With probability at least $1 - \delta$, $R(\hat{f}) \leq \varepsilon$ where

$$\varepsilon \geq \frac{\log |\bar{\mathcal{F}}| + \log(1/\delta)}{n}$$

(proof: exercise).

- We see that, through increasing n , improving our confidence is easier than increasing our accuracy.

PAC learnability

- Under the similar assumptions as the theorem above (binary classification, zero-one loss, $R(f^*) = 0$), and also assuming $\mathcal{X} = [0, 1]^p \dots$
- We say that an algorithm *PAC learns* \mathcal{F} using $\bar{\mathcal{F}}$ if, for any true $f \in \mathcal{F}$, and for any $0 < \varepsilon, \delta < 1$, the algorithm:
 - with probability at least $1 - \delta$, outputs a hypothesis $\hat{f} \in \bar{\mathcal{F}}$ with $R(\hat{f}) \leq \varepsilon$;
 - runs in polynomial time in $1/\varepsilon, 1/\delta, \text{size}(f)$ and p .
- Here \hat{f} is simply some chosen member of $\bar{\mathcal{F}}$, not necessarily chosen using ERM.
- We will not define $\text{size}(f)$ here
 - it is essentially the complexity of representing f .

Strong vs weak PAC learnability

- The above is sometimes known as a definition of *strong* PAC learning
 - to be satisfied, the algorithm needs to be able to achieve an arbitrarily small ε and δ .
- *Weak* PAC learning uses a fixed ε and δ .
- We say that an algorithm is a *weak PAC learning* algorithm if there exist fixed polynomials $p_\delta(\cdot, \cdot)$ and $p_\varepsilon(\cdot, \cdot)$ such that the algorithm outputs \hat{f} such that

$$R(\hat{f}) \leq \frac{1}{2} - \frac{1}{p_\varepsilon(p, \text{size}(f))}$$

with probability at least

$$\frac{1}{p_\delta(p, \text{size}(f))}.$$

Strong vs weak PAC learnability

- The definition of weak PAC learning is very weak
 - the algorithm only needs to be slightly better than a random guess!
- However:
 - a problem can be weak-learned if and only if it can be strong-learned!
- This insight is used in *boosting*, in which a weak PAC learning algorithm can be converted into a strong PAC learning algorithm
 - by boosting confidence
 - and accuracy.
- We will describe this approach later in the course.

Summary

- In a simple situation we can use a PAC bound to bound the excess risk.
- The bound depends on the
 - complexity of the model space
 - size of the training data.
- This bound did not depend on the true underlying f .
- We wish to obtain similar results more generally.

Further reading

- PRML chapter 7, page 344.

Concentration inequalities

ST420 Lecture 18

Richard Everitt

Recap

- We often fit a regression or classification model to training data using empirical risk minimisation.
- The excess risk $R(\hat{f}) - R(f^*)$ of \hat{f} tells us how good our predictions are using a model \hat{f} fit by ERM, compared to the optimal situation.
- If the approximation error were zero, we could bound the estimation error by bounding the generalisation error $R(f) - \hat{R}(f)$ of functions f .

Focus

- Recall the difference between $\hat{R}(f)$ and $R(f)$
 - $R(f)$ is the expectation of L over P
 - $\hat{R}(f)$ is an empirical average of L over P .
- Consider the case of a fixed f - this will be the case throughout the lecture.
- We already know something about the convergence of $\hat{R}(f)$ to $R(f)$.
- Recall the strong law of large numbers

$$\hat{R}(f) \rightarrow R(f)$$

almost surely as $n \rightarrow \infty$.

- We also have a central limit theorem.

Focus

- Issues:
 1. how fast is the convergence?
 2. can we say anything uniformly across $f \in \mathcal{F}$?
- For point 1, we need concentration inequalities
 - this lecture.
- For point 2, we will first consider the finite case, then the infinite
 - later.

Markov's inequality

- Lemma (Markov's inequality): Let $Z \geq 0$ be a random variable with $\mathbb{E}[Z] < \infty$. Then $\forall \varepsilon > 0$,

$$\mathbb{P}(Z \geq \varepsilon) \leq \frac{\mathbb{E}[Z]}{\varepsilon}.$$

- This bounds the probability of a non-negative random variable being in the tail
 - the bound is tighter the smaller the expectation.
- All loss functions L are non-negative, therefore we can take $Z = \hat{R}(f)$.
- Markov's inequality gives us

$$\mathbb{P}\left(\hat{R}(f) \geq \varepsilon\right) \leq \frac{\mathbb{E}\left[\hat{R}(f)\right]}{\varepsilon} = \frac{R(f)}{\varepsilon}.$$

- The bound on the probability of $\hat{R}(f)$ being large depends on $R(f)$.

Chebyshev's inequality

- Markov's inequality doesn't tell us much of any use when applied directly.
- However, we can use it to prove Chebyshev's inequality.
- **Lemma (Chebyshev's inequality):** Let Z be a random variable with $\mathbb{E}[Z], \mathbb{V}[Z] < \infty$. Then $\forall \varepsilon > 0$,

$$\mathbb{P}(|Z - \mathbb{E}[Z]| \geq \varepsilon) \leq \frac{\mathbb{V}[Z]}{\varepsilon^2}.$$

- This gives a quantitative bound on how far a random variable is from its mean, no matter which random variable is used!
 - the reason it is so general is that the variance is used in the bound
 - in this bound, the variance is governing how far the random variable can be from its mean.

Proof of Chebyshev's inequality

- We can use Markov's inequality to prove Chebyshev's inequality.

$$\begin{aligned}\mathbb{P}(|Z - \mathbb{E}[Z]| \geq \varepsilon) &= \mathbb{P}(|Z - \mathbb{E}[Z]|^2 \geq \varepsilon^2) \\ &= \mathbb{P}((Z - \mathbb{E}[Z])^2 \geq \varepsilon^2) \\ &\leq \frac{\mathbb{E}[(Z - \mathbb{E}[Z])^2]}{\varepsilon^2} \\ &= \frac{\mathbb{V}(Z)}{\varepsilon^2}\end{aligned}$$

Chebyshev's inequality for empirical risk

- Let $Z = \hat{R}(f)$, so that $\mathbb{E}[Z] = R(f)$.
- Also let $\sigma^2 = \mathbb{V}[L(Y, f(X))]$, so that

$$\begin{aligned}\mathbb{V}[\hat{R}(f)] &= \mathbb{V}\left[\frac{1}{n} \sum_{i=1}^n L(Y_i, f(X_i))\right] \\ &= \frac{1}{n^2} \sum_{i=1}^n \mathbb{V}[L(Y_i, f(X_i))] = \frac{\sigma^2}{n}.\end{aligned}$$

- Chebyshev's inequality gives us that $\forall \varepsilon > 0$,

$$\mathbb{P}\left(\left|\hat{R}(f) - R(f)\right| \geq \varepsilon\right) \leq \frac{\mathbb{V}[\hat{R}(f)]}{\varepsilon^2} = \frac{\sigma^2}{n\varepsilon^2} = \delta.$$

Chebyshev's inequality for empirical risk

- Notice that this proves the weak law of large numbers for the empirical risk
 - $\hat{R}(f)$ converges in probability to $R(f)$ as $n \rightarrow \infty$.
- It also gives us a (loose) bound on the rate that this occurs.
- We have a PAC-style bound on the generalisation error of a fixed f .

How loose is the bound?

- To get an idea of how loose the bound is, let's compare to the normal approximation from the central limit theorem.
- Under the CLT, what is the approximate distribution of $\hat{R}(f) - R(f)$?

$$\begin{aligned}\hat{R}(f) - R(f) &= \frac{1}{n} \sum_{i=1}^n L(Y_i, f(X_i)) - R(f) \\ &= \frac{\sigma}{\sqrt{n}} \left(\frac{1}{\sqrt{n}} \sum_{i=1}^n \frac{L(Y_i, f(X_i)) - R(f)}{\sigma} \right).\end{aligned}$$

- Thus $\hat{R}(f) - R(f)$ is approximately distributed as $\frac{\sigma}{\sqrt{n}} Z$, where $Z \sim \mathcal{N}(0, 1)$.

How loose is the bound?

- For a normal distribution, we have (stated without proof) that $\forall \varepsilon > 0$,

$$\mathbb{P}(|Z| \geq \varepsilon) \leq \exp\left(-\frac{\varepsilon^2}{2}\right).$$

- Therefore

$$\mathbb{P}\left(|Z| \geq \frac{\sqrt{n}\varepsilon}{\sigma}\right) \leq \exp\left(-\frac{n\varepsilon^2}{2\sigma^2}\right).$$

- Thus the following holds **approximately**

$$\mathbb{P}\left(\left|\hat{R}(f) - R(f)\right| \geq \varepsilon\right) \leq \exp\left(-\frac{n\varepsilon^2}{2\sigma^2}\right).$$

How loose is the bound?

- Chebyshev's inequality (**bound**)

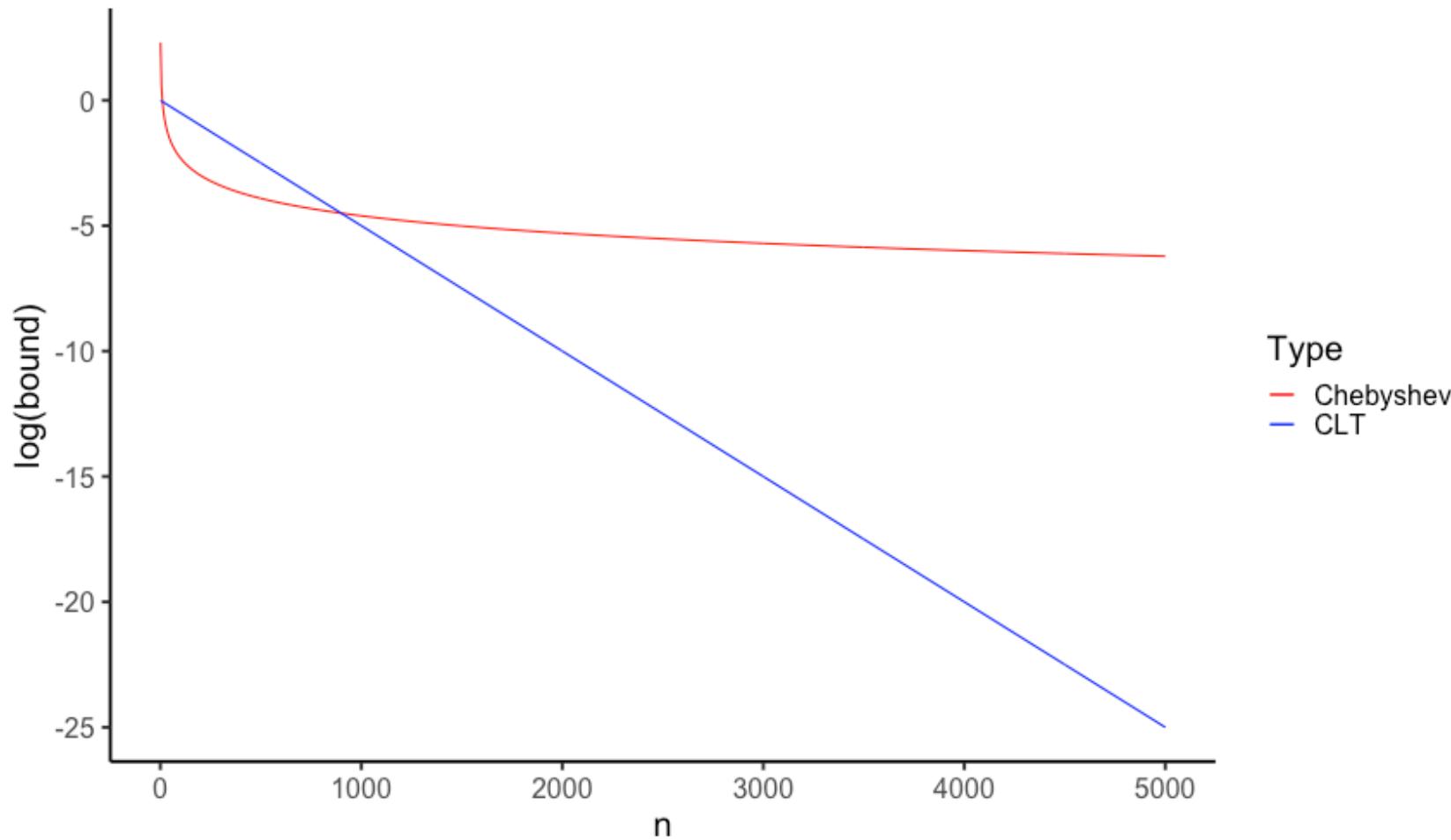
$$\mathbb{P}\left(\left|\hat{R}(f) - R(f)\right| \geq \varepsilon\right) \leq \frac{\sigma^2}{n\varepsilon^2}.$$

- CLT (**approximation**)

$$\mathbb{P}\left(\left|\hat{R}(f) - R(f)\right| \geq \varepsilon\right) \leq \exp\left(-\frac{n\varepsilon^2}{2\sigma^2}\right).$$

- The approximation converges exponentially fast - much faster than the Chebyshev bound.

How loose is the bound?



Chernoff bound

- Making use of a Chernoff bound can help us do better.
- **Theorem (Chernoff bound):** Let Z_1, \dots, Z_n be independent random variables and let $S_n = \sum_{i=1}^n Z_i$. Then $\forall \varepsilon > 0$,

$$\mathbb{P}(S_n \geq \varepsilon) \leq \inf_{s>0} \left\{ e^{-s\varepsilon} \prod_{i=1}^n \mathbb{E}[e^{sZ_i}] \right\}.$$

- This might be help us if we can bound $\mathbb{E}[e^{sZ_i}]$.

Proof of Chernoff bound

- Very similar to the proof of Chebyshev's inequality, but using the transformation $x \rightarrow e^{sx}$ instead of $x \rightarrow x^2$

$$\begin{aligned}\mathbb{P}(S_n \geq \varepsilon) &= \mathbb{P}\left(e^{sS_n} \geq e^{s\varepsilon}\right) \\ &\leq \frac{\mathbb{E}[e^{sS_n}]}{e^{s\varepsilon}} \\ &= e^{-s\varepsilon} \mathbb{E}\left[e^{s\sum_{i=1}^n Z_i}\right] \\ &= e^{-s\varepsilon} \mathbb{E}\left[\prod_{i=1}^n e^{sZ_i}\right] \\ &= e^{-s\varepsilon} \prod_{i=1}^n \mathbb{E}[e^{sZ_i}].\end{aligned}$$

Hoeffding's inequality

- We can bound $\mathbb{E}[e^{sZ_i}]$ if we can bound each Z_i . This leads to Hoeffding's inequality.
- **Theorem (Hoeffding's inequality):** Let Z_1, \dots, Z_n be independent random variables, each bounded by the interval $[a, b]$ and with the same expectation μ . Then Hoeffding's inequality says that for any $\varepsilon > 0$

$$\mathbb{P}\left(\left|\frac{1}{n} \sum_{i=1}^n Z_i - \mu\right| > \varepsilon\right) \leq 2 \exp\left(\frac{-2n\varepsilon^2}{(b-a)^2}\right).$$

Hoeffding's inequality

- When using zero-one loss, $L(Y, f(X))$ is a random variable that takes values in $\{0, 1\}$. Therefore we can use Hoeffding's inequality with $Z_i = L(Y_i, f(X_i))$, $\mu = R(f)$, $a = 0$ and $b = 1$, and obtain that for any $\varepsilon > 0$

$$\mathbb{P}\left(\left|\hat{R}(f) - R(f)\right| \geq \varepsilon\right) \leq 2 \exp(-2n\varepsilon^2).$$

Comparing the bounds

- Chebyshev's inequality (**bound**)

$$\mathbb{P}\left(\left|\hat{R}(f) - R(f)\right| \geq \varepsilon\right) \leq \frac{\sigma^2}{n\varepsilon^2}.$$

- Hoeffding's inequality (**bound**)

$$\mathbb{P}\left(\left|\hat{R}(f) - R(f)\right| \geq \varepsilon\right) \leq 2 \exp(-2n\varepsilon^2).$$

- CLT (**approximation**)

$$\mathbb{P}\left(\left|\hat{R}(f) - R(f)\right| \geq \varepsilon\right) \leq \exp\left(-\frac{n\varepsilon^2}{2\sigma^2}\right).$$

- CLT (**approximation**) for a classifier that guesses at random (so that $\sigma^2 = 1/4$)

$$\mathbb{P}\left(\left|\hat{R}(f) - R(f)\right| \geq \varepsilon\right) \leq \exp(-2n\varepsilon^2).$$

Summary

- We have found PAC-style bounds that hold quite generally
 - using a fixed f .
- The use of a fixed f restricts how useful this is
 - e.g. what is the effect of using different classes of functions?
- We will look into this next time.

From finite to infinite function spaces

ST420 Lecture 19

Richard Everitt

Recap

- We would like to bound the excess risk

$$R(\hat{f}) - R(f^*).$$

- At the moment we are focussing on bounding the size of the generalisation error $|\hat{R}(f) - R(f)|$ of functions f
 - this will help us bound the excess risk.
- We have been looking at PAC-style bounds, e.g. from the last lecture

$$\mathbb{P}(|\hat{R}(f) - R(f)| \geq \varepsilon) \leq 2 \exp(-2n\varepsilon^2).$$

- Last lecture: PAC-style bounds for $|\hat{R}(f) - R(f)|$ for a fixed function f .

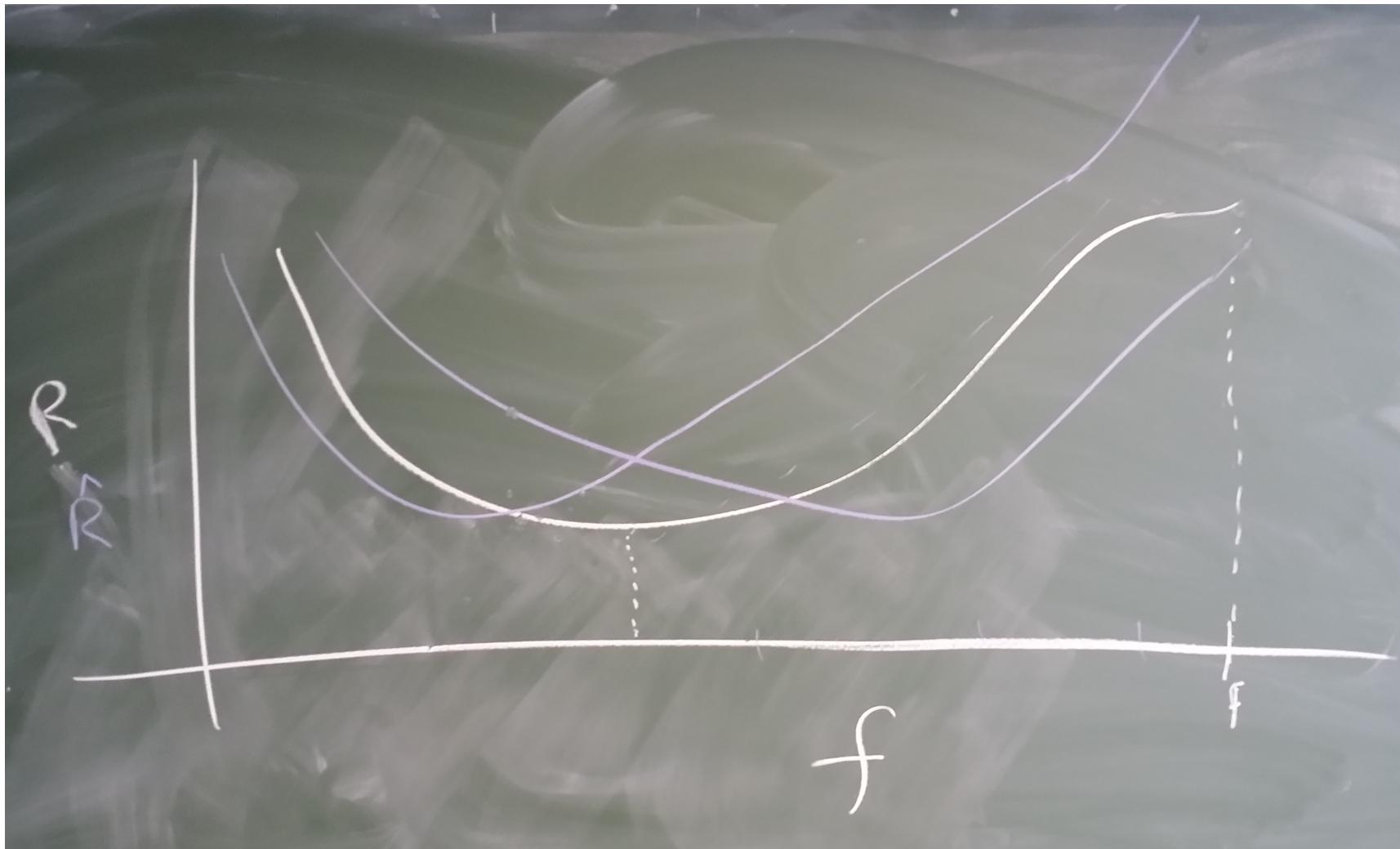
Limitation of bounds for a fixed f

- As an example, take the bound from Hoeffding's inequality from the last lecture from the last lecture

$$\mathbb{P}\left(\left|\hat{R}(f) - R(f)\right| \geq \varepsilon\right) \leq 2 \exp(-2n\varepsilon^2) = \delta.$$

- Let's think more closely about what this is saying.
- The randomness is over different training data sets
 - for some f , the proportion of training data sets that result in the size of the generalisation error being less than ε is greater than $1 - \delta$
 - however, the training data sets that result in a small risk for function f are not necessarily the same as those that result in a small risk for a different function.
- We would like to obtain a small risk **simultaneously** across **all** functions.

Limitation of bounds for a fixed f : picture



Brute force approach

- Insist that $|\hat{R}(f) - R(f)|$ is small for all functions
 - find a bound on $\sup_{f \in \bar{\mathcal{F}}} |\hat{R}(f) - R(f)|$.
- This is called a *uniform bound* across $\bar{\mathcal{F}}$.
- To begin, we look at the case where $\bar{\mathcal{F}}$ is a finite set.
- We number each of the functions in the set as $f_1, \dots, f_{|\bar{\mathcal{F}}|}$.
- Idea: to explicitly write down the training data sets for which the bound fails for each function.

Deriving a uniform bound

- Let B_k be the "bad" set of sets of training data for which the generalisation error is bigger than ε for function f_k

$$B_k = \left\{ \mathcal{T} \mid |\hat{R}(f_k) - R(f_k)| \geq \varepsilon \right\}.$$

- By Hoeffding's inequality,

$$\mathbb{P}(B_k) \leq \delta.$$

- Now using the union bound we have

$$\begin{aligned} \mathbb{P}(B_1 \cup \dots \cup B_{|\bar{\mathcal{F}}|}) &\leq \sum_{k=1}^{|\bar{\mathcal{F}}|} \mathbb{P}(B_k) \\ &\leq |\bar{\mathcal{F}}| \delta. \end{aligned}$$

Uniform bound

- The result is

$$\begin{aligned}\mathbb{P}\left(\exists f \in \bar{\mathcal{F}} \mid |\hat{R}(f) - R(f)| \geq \varepsilon\right) &\leq |\bar{\mathcal{F}}| \delta \\ &\leq 2|\bar{\mathcal{F}}| \exp(-2n\varepsilon^2).\end{aligned}$$

- $1 - \mathbb{P}\left(\exists f \in \bar{\mathcal{F}} \mid |\hat{R}(f) - R(f)| \geq \varepsilon\right)$ is the probability that the size of the generalisation error $< \varepsilon$ for all functions in the set.

Fixed f compared to uniform bound

- We rewrite the bounds as we did when we first introduced PAC bounds.
- For fixed f
 - for all $\delta > 0$, with probability at least $1 - \delta$, for a fixed $f \in \bar{\mathcal{F}}$,

$$|\hat{R}(f) - R(f)| \leq \sqrt{\frac{\log \frac{2}{\delta}}{2n}}.$$

- Uniform bound
 - for all $\delta > 0$, with probability at least $1 - \delta$,

$$\sup_{f \in \bar{\mathcal{F}}} |\hat{R}(f) - R(f)| \leq \sqrt{\frac{\log |\bar{\mathcal{F}}| + \log \frac{2}{\delta}}{2n}}.$$

Uniform bound: discussion

- For all $\delta > 0$, with probability at least $1 - \delta$

$$\sup_{f \in \bar{\mathcal{F}}} |\hat{R}(f) - R(f)| \leq \sqrt{\frac{\log |\bar{\mathcal{F}}| + \log \frac{2}{\delta}}{2n}}.$$

- As n increases, the generalisation error goes to 0
 - aside: because of the way we have rewritten the bound, we can see that it is expensive in terms of how we must increase n in order to improve accuracy.
- An increase in the size $|\bar{\mathcal{F}}|$ of the function class, which we can think of as being a measure of the flexibility of the function class, leads to a decrease in accuracy.

Infinite function spaces

- How do we obtain something similar when the function space is infinite?
- We omit material relating only to the countably infinite case, and move straight to the uncountably infinite.
- Before we do so, we will make a slight alteration to the quantity we are looking at
 - we have been studying $|\hat{R}(f) - R(f)|$ (mainly because we started with Chebyshev's inequality, and have been comparing our bounds with that one)
 - we will instead look at $R(f) - \hat{R}(f)$.
- Instead we have that for all $\delta > 0$, for a fixed $f \in \bar{\mathcal{F}}$,

$$\mathbb{P}\left(R(f) - \hat{R}(f) \geq \sqrt{\frac{\log \frac{1}{\delta}}{2n}}\right) \leq \delta.$$

Uncountably infinite function spaces

- As we have been doing already, we restrict our attention to the binary classification problem.
- Idea: although the function space is infinite, we have a finite set of training points.
- We look at the "projection" of the function class onto the X points in a set of training data, defined to be

$$\bar{\mathcal{F}}_X := \left\{ (f(X_1), \dots, f(X_n)) \mid f \in \bar{\mathcal{F}} \right\}.$$

Shattered sets

- Each member of the set is a vector of length n , with entry being either 0 or 1.
- The maximum size of the set $\bar{\mathcal{F}}_X$ is 2^n .
- If

$$|\bar{\mathcal{F}}_X| = 2^n,$$

we say that X has been *shattered* by the class of functions $\bar{\mathcal{F}}$.

- The size of $\bar{\mathcal{F}}_X$ will be smaller than this if not all possible classifications can be made by $f \in \bar{\mathcal{F}}$.

Growth function

- The *growth function* is the maximum (over possible samples) number of different classifications we may obtain when using function class $\bar{\mathcal{F}}$.
- Formally

$$S_{\bar{\mathcal{F}}}(n) = \sup_X |\bar{\mathcal{F}}_X|$$

where the supremum is over sets X of size n .

- This gives an indication of the size/flexibility of the class $\bar{\mathcal{F}}$
 - if we can obtain more classifications, the class is more flexible, so the value taken by the function will tend to be larger.

VC dimension

- The *VC dimension* is the dimensionality of the largest set that is shattered by $\bar{\mathcal{F}}$.
- Named after Vapnik and Chervonenkis.
- It is the largest n such that

$$S_{\bar{\mathcal{F}}}(n) = 2^n$$

- increase the size of the data set as far as we can whilst still achieving perfect classifications.
- The definitions on the previous few slides turn out to be useful for measuring the flexibility of infinite function classes $\bar{\mathcal{F}}$ when giving bounds on generalisation error.

Bound using growth function

- Theorem (bound using growth function): For all $\delta > 0$, with probability at least $1 - \delta$,

$$\sup_{f \in \bar{\mathcal{F}}} (R(f) - \hat{R}(f)) \leq 2 \sqrt{2 \frac{\log S_{\bar{\mathcal{F}}}(2n) + \log \frac{2}{\delta}}{n}}.$$

- Note the similarity to the bound for the finite case
 - $S_{\bar{\mathcal{F}}}(2n)$ is playing the role of measuring the flexibility of the function class.

Sauer's theorem

- Theorem (Sauer's theorem): Let $\bar{\mathcal{F}}$ have VC dimension d . Then

1. $\forall n,$

$$S_{\bar{\mathcal{F}}}(n) \leq \sum_{i=0}^d \binom{n}{i},$$

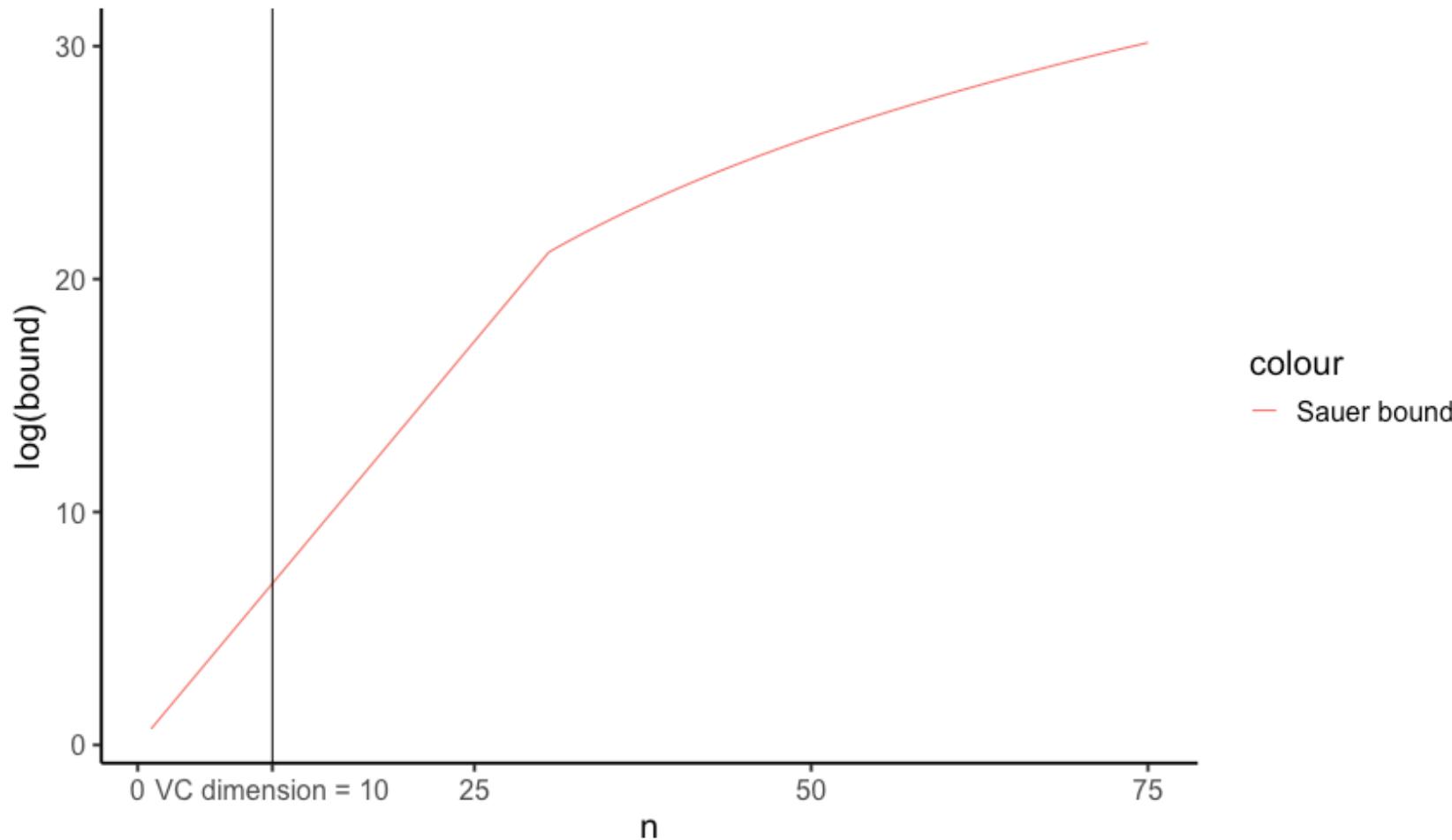
2. for $n \geq d$,

$$S_{\bar{\mathcal{F}}}(n) \leq \left(\frac{en}{d} \right)^d.$$

Sauer's theorem

- This highlights the importance of VC dimension
 - for sets smaller than the VC dimension, the growth function grows exponentially with n
 - for sets larger than the VC dimension, the growth function grows only polynomially with n .
- Point 1 is not as interesting as point 2, since for $n \leq d$ we already know that $S_{\bar{F}}(n) = 2^n$.
- However, there will be values of n for which the bound in point 1 is tighter than that in point 2.

Bound on the growth function from Sauer's theorem



Corollary

- **Corollary (VC bound):** For all $\delta > 0$, with probability at least $1 - \delta$, if $\bar{\mathcal{F}}$ has VC dimension d , for $n \geq d$,

$$\sup_{f \in \bar{\mathcal{F}}} (R(f) - \hat{R}(f)) \leq 2 \sqrt{2 \frac{d \log\left(\frac{2en}{d}\right) + \log \frac{2}{\delta}}{n}}.$$

- This is a simple corollary of the previous two theorems
 - the VC dimension is being used to measure the flexibility of the function class.

What now?

- Next lecture we shall explore the concept of VC dimension in a little more depth.
- We will then prove some of these results.
- Following this we will discuss how these bounds may be used in practice.

Further reading

- ESL chapter 7.

VC dimension continued

ST420 Lecture 20

Richard Everitt

Recap

- We would like to bound the excess risk

$$R(\hat{f}) - R(f^*).$$

- We have been focussing on bounding the generalisation error $R(f) - \hat{R}(f)$ of functions f
 - this will help us bound the excess risk.
- We have been looking at PAC-style bounds, e.g.

$$\mathbb{P}(|\hat{R}(f) - R(f)| \geq \varepsilon) \leq 2 \exp(-2n\varepsilon^2).$$

- Last lecture: PAC-style bounds for $R(f) - \hat{R}(f)$ across a finite or infinite function class.

Recap

- Look at the "projection" of the function class onto the X points in a set of training data, defined to be

$$\bar{\mathcal{F}}_X := \left\{ (f(X_1), \dots, f(X_n)) \mid f \in \bar{\mathcal{F}} \right\}.$$

- If

$$|\bar{\mathcal{F}}_X| = 2^n,$$

we say that X has been *shattered* by the class of functions $\bar{\mathcal{F}}$.

- The *VC dimension* is the dimensionality of the largest set that is shattered by $\bar{\mathcal{F}}$.
- Growth function:

$$S_{\bar{\mathcal{F}}}(n) = \sup_X |\bar{\mathcal{F}}_X|.$$

VC bound

- **Theorem (bound using growth function):** For all $\delta > 0$, with probability at least $1 - \delta$,

$$\sup_{f \in \bar{\mathcal{F}}} (R(f) - \hat{R}(f)) \leq 2 \sqrt{2 \frac{\log S_{\bar{\mathcal{F}}}(2n) + \log \frac{2}{\delta}}{n}}.$$

- **Corollary (VC bound):** For all $\delta > 0$, with probability at least $1 - \delta$, if $\bar{\mathcal{F}}$ has VC dimension d , for $n \geq d$,

$$\sup_{f \in \bar{\mathcal{F}}} (R(f) - \hat{R}(f)) \leq 2 \sqrt{2 \frac{d \log\left(\frac{2en}{d}\right) + \log \frac{2}{\delta}}{n}}.$$

Example: splitting the real line

- Let $\mathcal{X} = \mathbb{R}$ and $\mathcal{Y} = \{0, 1\}$.
- Let

$$\bar{\mathcal{F}} = \left\{ f : \mathcal{X} \rightarrow \mathcal{Y} \mid f(x) = I(x \geq \beta^0), \beta^0 \in \mathbb{R} \right\}.$$

- For $n = 1$ there are two possible classifications of the point, so $S_{\bar{\mathcal{F}}}(1) = 2$.
- For $n = 2$ there are three possible classifications of the two points - suppose that (x_1, x_2) are in order of increasing size
 - then if $\beta^0 < x_1 < x_2$, $(f(x_1), f(x_2)) = (1, 1)$,
 - if $x_1 < \beta^0 < x_2$, $(f(x_1), f(x_2)) = (0, 1)$,
 - and if $x_1 < x_2 < \beta^0$, $(f(x_1), f(x_2)) = (0, 0)$,
 - therefore $S_{\bar{\mathcal{F}}}(2) = 3$.

Example: splitting the real line

- In general, having n points partitions \mathcal{X} into $n + 1$ pieces
- When β^0 falls into each of these different pieces, we obtain a different classification.
- Therefore $S_{\bar{\mathcal{F}}}(n) = n + 1$.
- When can the set be shattered?
 - Only when $n = 1$.
 - Therefore the VC dimension of $\bar{\mathcal{F}}$ is 1.

Example: line in two dimensions

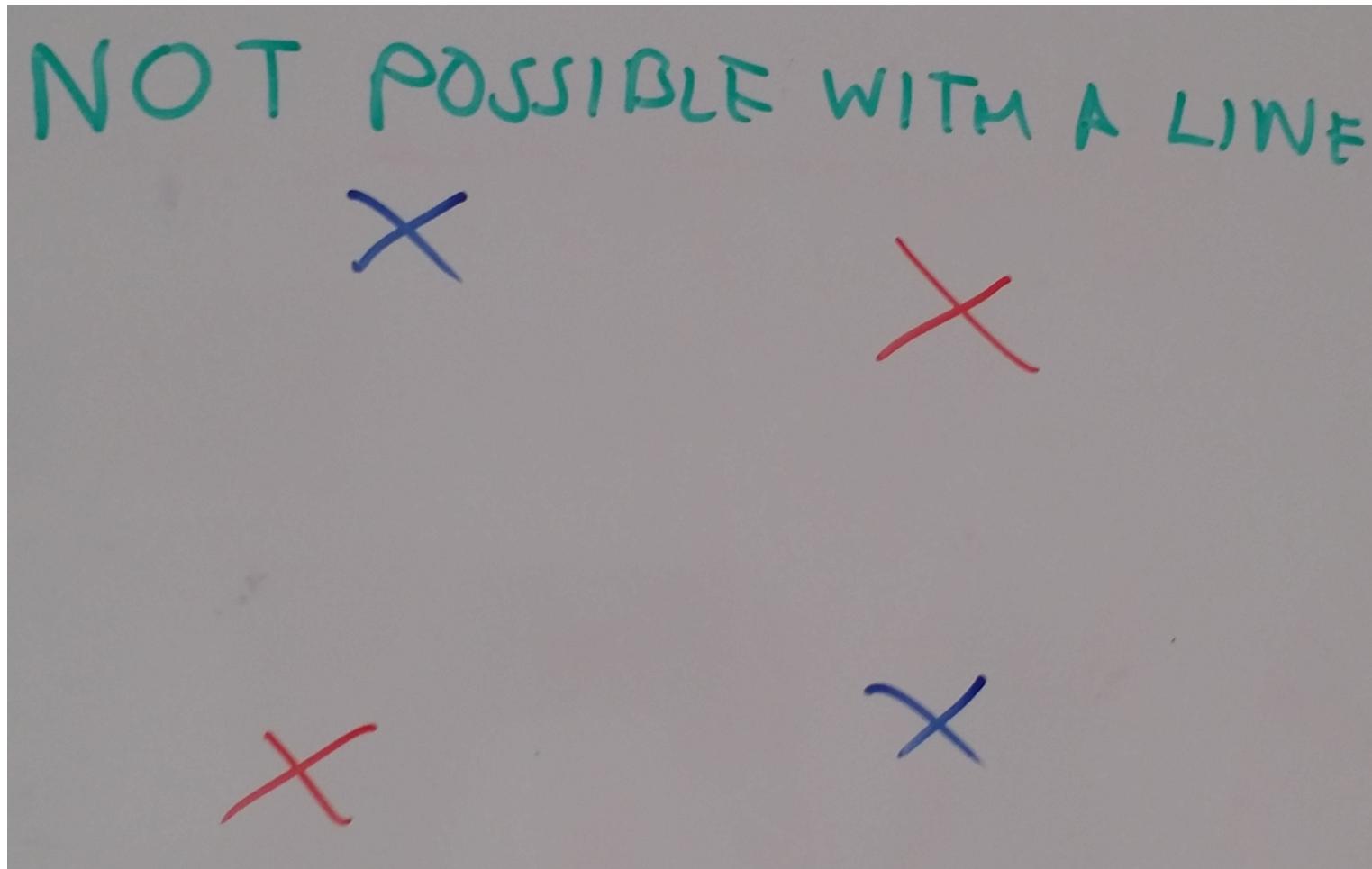
- Let $\mathcal{X} = \mathbb{R}^2$ and $\mathcal{Y} = \{0, 1\}$.
- Let

$$\bar{\mathcal{F}} = \left\{ f : \mathcal{X} \rightarrow \mathcal{Y} \mid f(x) = I(\beta^1 x^1 + \beta^2 x^2 + \beta^0 \geq 0), \beta^0, \beta^1 \in \mathbb{R} \right\}.$$

- For $n = 1$ there are two possible classifications of the point, so $S_{\bar{\mathcal{F}}}(1) = 2$.
- For $n = 2$ all classifications are possible, so $S_{\bar{\mathcal{F}}}(2) = 4$.
- For $n = 3$ all classifications are possible, so $S_{\bar{\mathcal{F}}}(3) = 8$.
- For $n = 4$ not all classifications are possible (see picture).

Example: line in two dimensions

- For $n = 4$ not all classifications are possible, thus the VC dimension is 3.



Example: linear classifier in general

- Let $\mathcal{X} = \mathbb{R}^p$ and $\mathcal{Y} = \{0, 1\}$.
- Let

$$\bar{\mathcal{F}} = \left\{ f : \mathcal{X} \rightarrow \mathcal{Y} \mid f(x) = \text{sign}(X^T \beta + \beta^0), \beta \in \mathbb{R}^p, \beta^0 \in \mathbb{R} \right\}.$$

- Let \mathbf{X} be the design matrix (including intercept terms) with dimension $n \times (p + 1)$.
- **Lemma:** A set $\{x_1, \dots, x_n\}$ can be shattered if and only if the rows of \mathbf{X} are linearly independent.
- Using this lemma, we see that, since the rows of \mathbf{X} can be linearly independent for at most $p + 1$ points, we have that the VC dimension of $\bar{\mathcal{F}}$ is $p + 1$.

Proof of lemma

- If:
- Assume the rows of \mathbf{X} are linearly independent.
- Then $\text{rank}(\mathbf{X}) = n$, then $\mathbf{X} : \mathbb{R}^{p+1} \rightarrow \mathbb{R}^n$ is surjective.
- This means that all classification decisions are possible, which means that the set $\{x_1, \dots, x_n\}$ can be shattered.

Proof of lemma

- Only if:
- Assume we can shatter $\{x_1, \dots, x_n\}$.
- Then we can find $\beta' = (\beta^0, \beta^T)^T \in \mathbb{R}^{p+1}$ such that $\mathbf{X}\beta'$ lies in any of the 2^n orthants of \mathbb{R}^n - i.e. we can obtain any of the possible classification decisions. Thus, these 2^n points are found in the range of \mathbf{X} .
- The span of these 2^n points is \mathbb{R}^n . Therefore $\mathbb{R}^n \subseteq \text{range}(\mathbf{X})$.
- Therefore $\text{rank}(\mathbf{X}) = n$, thus the rows of \mathbf{X} can be linearly independent.

Rectangles

- Let $\mathcal{X} = \mathbb{R}^2$ and $\mathcal{Y} = \{0, 1\}$.

- Let

$$\bar{\mathcal{F}} = \{f : \mathcal{X} \rightarrow \mathcal{Y} \mid f(x) = I(x \in A), A = [a, b] \times [c, d], \quad a, b, c, d \in \mathbb{R}\}.$$

- We will not prove this here, but using pictures, we can see that the VC dimension is 4.

How to find the VC dimension

- **Easy part:** Show that for $n \leq d$ there exists a set of n points that can be shattered by the class.
- **Difficult part:** Show that there is no set of $n > d$ points that can be shattered.

Number of parameters?

- In all of the examples we have seen so far, the VC dimension is equal to the number of parameters used to specify $\bar{\mathcal{F}}$.
- Is this true in general?
 - No!
- Counter example:
 - $\mathcal{X} = \mathbb{R}$, $\mathcal{Y} = \{0, 1\}$ and $\bar{\mathcal{F}} = \{f : \mathcal{X} \rightarrow \mathcal{Y} \mid f(x) = \text{sign}(\sin(\beta x)), \beta \in \mathbb{R}\}$
 - here there is only one parameter, but the VC dimension is infinite.

VC dimension of popular methods

- For neural networks
 - the VC dimension is related to the number of edges in the graph representing its structure
 - reassuring result, since this is what we might intuitively expect.
- For support vector machines
 - the VC dimension is less than or equal to
$$1 + 1/\text{margin}^2.$$
- Recall that SVMs were originally motivated by statistical learning theory
 - this is not the only method that has been inspired by the theory.

Revisiting bounds

- **Theorem (bound using growth function):** For all $\delta > 0$, with probability at least $1 - \delta$,

$$\sup_{f \in \bar{\mathcal{F}}} (R(f) - \hat{R}(f)) \leq 2 \sqrt{2 \frac{\log S_{\bar{\mathcal{F}}}(2n) + \log \frac{2}{\delta}}{n}}.$$

- To prove this theorem, we need a result that uses the idea of *symmetrisation*
 - this is the idea of replacing the true risk with the empirical risk calculated using a "ghost" sample
 - this is a mathematical construct, rather than something we do in practice.

Symmetrisation lemma

- Lemma (symmetrisation): For all $\varepsilon > 0$, such that $n\varepsilon^2 \geq 2$,

$$\mathbb{P}\left(\sup_{f \in \bar{\mathcal{F}}} R(f) - \hat{R}(f) \geq \varepsilon\right) \leq 2\mathbb{P}\left(\sup_{f \in \bar{\mathcal{F}}} \hat{R}'(f) - \hat{R}(f) \geq \varepsilon/2\right),$$

where \hat{R}' is estimated using the ghost training sample \mathcal{T}' .

- We will now use this result, with Hoeffding's inequality, to prove the bound on the previous slide (next time).
- First will prove the symmetrisation lemma.

Proof of symmetrisation lemma

- Let f_n be the function achieving the supremum. Then

$$\begin{aligned} I(R(f_n) - \hat{R}(f_n) > \varepsilon) I(R(f_n) - \hat{R}'(f_n) < \varepsilon/2) &= I(R(f_n) - \hat{R}(f_n) > \varepsilon \wedge \\ &\quad \hat{R}'(f_n) - R(f_n) \geq -\varepsilon/2) \\ &\leq I(\hat{R}'(f_n) - \hat{R}(f_n) > \varepsilon/2). \end{aligned}$$

- Taking expectations with respect to \mathcal{T}' gives

$$I(R(f_n) - \hat{R}(f_n) > \varepsilon) \mathbb{P}(R(f_n) - \hat{R}'(f_n) < \varepsilon/2) \leq \mathbb{P}(\hat{R}'(f_n) - \hat{R}(f_n) > \varepsilon/2).$$

Proof of symmetrisation lemma

- The loss is a random variable that takes values in $[0, 1]$, thus its variance is $\leq 1/4$ (by Popoviciu's inequality on variance).
- Using Chebyshev's inequality we obtain

$$\mathbb{P}\left(R(f_n) - \hat{R}'(f_n) \geq \varepsilon/2\right) \leq \frac{1}{n\varepsilon^2} .$$

- This gives us

$$I\left(R(f_n) - \hat{R}(f_n) > \varepsilon\right)\left(1 - \frac{1}{n\varepsilon^2}\right) \leq \mathbb{P}\left(\hat{R}'(f_n) - \hat{R}(f_n) > \varepsilon/2\right).$$

- Taking expectations with respect to \mathcal{T} gives

$$\mathbb{P}\left(R(f_n) - \hat{R}(f_n) > \varepsilon\right)\left(1 - \frac{1}{n\varepsilon^2}\right) \leq \mathbb{P}\left(\hat{R}'(f_n) - \hat{R}(f_n) > \varepsilon/2\right).$$

- Then using $n\varepsilon^2 \geq 2$, we obtain the result.

Next time

- We will use the symmetrisation lemma, with Hoeffding's inequality, to prove the growth function bound.
- We will then look at Sauer's theorem, and how it leads to the bound that uses the VC dimension.
- Finally we will revisit the idea of structural risk minimisation.

Further reading

- ESL chapter 7.

VC dimension and structural risk minimisation

ST420 Lecture 21

Richard Everitt

Recap

- We would like to bound the excess risk

$$R(\hat{f}) - R(f^*).$$

- We have been focussing on bounding the generalisation error $R(f) - \hat{R}(f)$ of functions f
 - this will help us bound the excess risk.
- We have been looking at PAC-style bounds, e.g.

$$\mathbb{P}(|\hat{R}(f) - R(f)| \geq \varepsilon) \leq 2 \exp(-2n\varepsilon^2).$$

- Last lecture: PAC-style bounds for $R(f) - \hat{R}(f)$ across a finite, countably infinite or uncountably infinite function class.

Recap

- Look at the "projection" of the function class onto the X points in a set of training data, defined to be

$$\bar{\mathcal{F}}_X := \left\{ (f(X_1), \dots, f(X_n)) \mid f \in \bar{\mathcal{F}} \right\}.$$

- If

$$|\bar{\mathcal{F}}_X| = 2^n,$$

we say that X has been *shattered* by the class of functions $\bar{\mathcal{F}}$.

- The *VC dimension* is the dimensionality of the largest set that is shattered by $\bar{\mathcal{F}}$.
- Growth function:

$$S_{\bar{\mathcal{F}}}(n) = \sup_X |\bar{\mathcal{F}}_X|.$$

VC bound

- **Theorem (bound using growth function):** For all $\delta > 0$, with probability at least $1 - \delta$,

$$\sup_{f \in \bar{\mathcal{F}}} (R(f) - \hat{R}(f)) \leq 2 \sqrt{2 \frac{\log S_{\bar{\mathcal{F}}}(2n) + \log \frac{2}{\delta}}{n}}.$$

- **Corollary (VC bound):** For all $\delta > 0$, with probability at least $1 - \delta$, if $\bar{\mathcal{F}}$ has VC dimension d , for $n \geq d$,

$$\sup_{f \in \bar{\mathcal{F}}} (R(f) - \hat{R}(f)) \leq 2 \sqrt{2 \frac{d \log\left(\frac{2en}{d}\right) + \log \frac{2}{\delta}}{n}}.$$

Symmetrisation lemma

- Lemma (symmetrisation): For all $\varepsilon > 0$, such that $n\varepsilon^2 \geq 2$,

$$\mathbb{P}\left(\sup_{f \in \bar{\mathcal{F}}} R(f) - \hat{R}(f) \geq \varepsilon\right) \leq 2\mathbb{P}\left(\sup_{f \in \bar{\mathcal{F}}} \hat{R}'(f) - \hat{R}(f) \geq \varepsilon/2\right),$$

where \hat{R}' is estimated using the ghost training sample \mathcal{T}' .

- We will now use this result to prove the growth function bound. We use the following form of Hoeffding's inequality

$$\mathbb{P}\left(\left(\hat{R}(f) - \hat{R}'(f)\right) > \varepsilon\right) \leq \exp(-n\varepsilon^2/2).$$

Proof of growth function bound

$$\begin{aligned}\mathbb{P}\left(\sup_{f \in \bar{\mathcal{F}}}\left(R(f) - \hat{R}(f)\right) \geq \varepsilon\right) &\leq 2\mathbb{P}\left(\sup_{f \in \bar{\mathcal{F}}}\left(\hat{R}'(f) - \hat{R}(f)\right) \geq \varepsilon/2\right) \\ &= 2\mathbb{P}\left(\sup_{f \in \bar{\mathcal{F}}_{X,X'}}\left(\hat{R}'(f) - \hat{R}(f)\right) \geq \varepsilon/2\right) \\ &\leq 2S_{\bar{\mathcal{F}}}(2n)\mathbb{P}\left(\left(\hat{R}'(f) - \hat{R}(f)\right) \geq \varepsilon/2\right) \\ &\leq 2S_{\bar{\mathcal{F}}}(2n)\exp(-n\varepsilon^2/8).\end{aligned}$$

Then, as usual, we may rewrite this expression as the one stated in the theorem.

Sauer's theorem

- Theorem (Sauer's theorem): Let $\bar{\mathcal{F}}$ have VC dimension d . Then

1. $\forall n,$

$$S_{\bar{\mathcal{F}}}(n) \leq \sum_{i=0}^d \binom{n}{i},$$

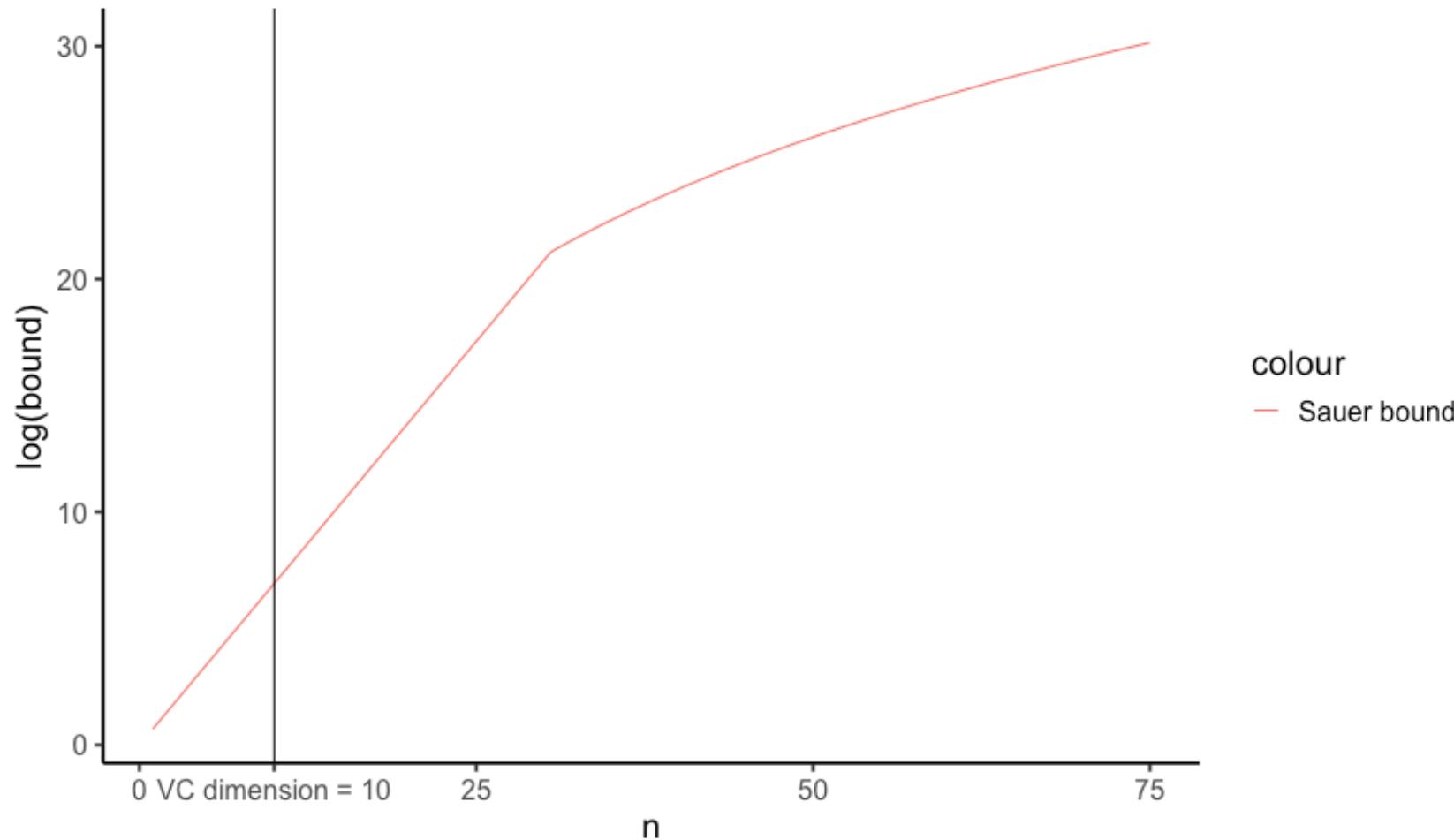
2. for $n \geq d$,

$$S_{\bar{\mathcal{F}}}(n) \leq \left(\frac{en}{d} \right)^d.$$

Sauer's theorem

- This highlights the importance of VC dimension
 - for sets smaller than the VC dimension, the growth function grows exponentially with n
 - for sets larger than the VC dimension, the growth function grows only polynomially with n .
- Point 1 is not as interesting as point 2, since for $n \leq d$ we already know that $S_{\bar{\mathcal{F}}}(n) = 2^n$.
- However, there will be values of n for which the bound in point 1 is tighter than that in point 2.

Bound on the growth function from Sauer's theorem



Corollary

- **Corollary (VC bound):** For all $\delta > 0$, with probability at least $1 - \delta$, if $\bar{\mathcal{F}}$ has VC dimension d , for $n \geq d$,

$$\sup_{f \in \bar{\mathcal{F}}} (R(f) - \hat{R}(f)) \leq 2 \sqrt{2 \frac{d \log\left(\frac{2en}{d}\right) + \log \frac{2}{\delta}}{n}}.$$

- This is a simple corollary of the previous two theorems
 - the VC dimension is being used to measure the flexibility of the function class.

VC entropy, fat shattering dimension

- A feature of what we have presented is that these bounds are not dependent on the underlying distribution P
 - could be an advantage?
 - but maybe the bound is looser since it needs to be true for all P
 - the *VC entropy* is an alternative that depends on P
 - leads to similar bound, but where the VC entropy takes the place of the VC dimension.
- *Fat shattering dimension* allows us to extend to the regression setting similar notions of the flexibility of a function class.

Rademacher complexity

- The *empirical Rademacher complexity* of a function class $\bar{\mathcal{F}}$ is defined to be

$$\hat{\mathcal{R}}(\bar{\mathcal{F}}) = \mathbb{E}_S \left[\sup_{f \in \bar{\mathcal{F}}} \frac{1}{n} \sum_{i=1}^n S_i L(Y_i, f(X_i)) \right],$$

where the S_i are Rademacher random variables (uniform on $\{-1, 1\}$) and the expectation is over the S .

- The *Rademacher complexity* $\mathcal{R}(\bar{\mathcal{F}})$ is defined to be the expectation over training data of the empirical Rademacher complexity
 - the empirical Rademacher complexity is random (depends on the training set)
 - the Rademacher complexity is deterministic.
- The Rademacher complexity is a measure of how well the function class fits random noise.
- The empirical Rademacher complexity depends on the underlying P (as did the VC entropy).

Rademacher complexity bound

- Theorem (Rademacher bound): For all $\delta > 0$, with probability at least $1 - \delta$,

$$\sup_{f \in \bar{\mathcal{F}}} (R(f) - \hat{R}(f)) \leq 2\mathcal{R}(\bar{\mathcal{F}}) + \sqrt{\frac{\log \frac{1}{\delta}}{2n}}.$$

- Theorem (Empirical Rademacher bound): For all $\delta > 0$, with probability at least $1 - \delta$,

$$\sup_{f \in \bar{\mathcal{F}}} (R(f) - \hat{R}(f)) \leq 2\hat{\mathcal{R}}(\bar{\mathcal{F}}) + \sqrt{2 \frac{\log \frac{2}{\delta}}{n}}.$$

- Also, it turns out that for VC dimension d , for some C we have

$$\hat{\mathcal{R}}(\bar{\mathcal{F}}) \leq C \sqrt{\frac{d}{n}}.$$

Returning to excess risk

- Recall that we are interested in the excess risk for comparing the prediction error of our ERM estimate to the optimal

$$R(\hat{f}) - R(f^*) = \underbrace{(R(\hat{f}) - R(\bar{f}^*))}_{\text{estimation error}} + \underbrace{(R(\bar{f}^*) - R(f^*))}_{\text{approximation error}}.$$

- Estimation error:

$$R(\hat{f}) - R(\bar{f}^*) \leq (R(\hat{f}) - \hat{R}(\hat{f})) + (\hat{R}(\bar{f}^*) - R(\bar{f}^*)).$$

- We have found bounds for $R(f) - \hat{R}(f)$ over **all** f . Therefore we can bound the estimation error. For example, in the finite case we have

$$\begin{aligned} R(\hat{f}) - R(\bar{f}^*) &\leq |R(\hat{f}) - \hat{R}(\hat{f})| + |\hat{R}(\bar{f}^*) - R(\bar{f}^*)| \\ &\leq 2\sqrt{\frac{\log |\bar{\mathcal{F}}| + \log \frac{2}{\delta}}{2n}}. \end{aligned}$$

Review: how is this theory useful?

- They give a theoretical basis for our understanding of supervised classification, e.g. about:
 - the important role played by generalising from training data;
 - how the accuracy of predictions, and the confidence we place in them, changes with the size n of our training data
 - the impact of the flexibility/restrictiveness of the class of functions we use.
- It has provided the inspiration for new methods
 - support vector machine
 - boosting (described next week).
- It can help us perform model selection
 - use bounds in structural risk minimisation (SRM).

Combatting overfitting II: structural risk minimisation

- Structural risk minimisation (SRM) is the idea of choosing from an infinite sequence of models $\bar{\mathcal{F}}_1, \bar{\mathcal{F}}_2, \dots$, of increasing size, using the empirical risk, and an added penalty for the complexity of the model, i.e.

$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}_d, d \in \mathbb{N}} \hat{R}(f) + \text{pen}(d, n).$$

- What is $\text{pen}(d, n)$?
 - penalises models as d gets larger
 - can be offset by n being large.
- Ideally we would choose $\text{pen}(d, n)$ to be the generalisation error
 - not available directly, but we can bound the generalisation error, and use this bound as $\text{pen}(d, n)$.

What could we use as the penalty?

- Consider, as an example, the case where $\bar{\mathcal{F}}$ is finite.
- We know that for any f ,

$$R(f) - \hat{R}(f) \leq \sqrt{\frac{\log |\bar{\mathcal{F}}| + \log \frac{1}{\delta}}{2n}}.$$

- So

$$R(f) \leq \hat{R}(f) + \sqrt{\frac{\log |\bar{\mathcal{F}}| + \log \frac{1}{\delta}}{2n}}.$$

- This bound may be loose, but can still be useful in practice, so we could take, for some δ ,

$$\text{pen}(|\bar{\mathcal{F}}|, n) = \sqrt{\frac{\log |\bar{\mathcal{F}}| + \log \frac{1}{\delta}}{2n}}.$$

Next lecture

- We examine two alternative approaches for estimating R
 - cross-validation
 - bootstrapping.
- Both are based on *resampling* the observed data.

Cross validation and bootstrapping

ST420 Lecture 22

Richard Everitt

Recap

- We would like to bound the excess risk

$$R(\hat{f}) - R(f^*).$$

- We have been focussing on bounding the generalisation error $R(f) - \hat{R}(f)$ of functions f
 - we have been looking at PAC-style bounds.
- This lecture: different approaches to estimating $R(\hat{f})$, based on resampling
 - cross validation
 - bootstrapping.

Using training data to estimate risk of \hat{f}

- We fit a function \hat{f} using ERM (or some other approach).
- If we used ERM, \hat{f} is the function that minimised the empirical risk $\hat{R}(f)$.
- The bounds we have found help us to understand the degree to which this is a reasonable proxy for using the true risk $R(f)$.
- However, we know that $\hat{R}(\hat{f})$ will tend to be smaller than $R(\hat{f})$ since \hat{f} is chosen to closely match the training data, and will tend to match this more closely than a different random sample from P .
- In lecture 4 we studied the in-sample error, and found that the optimism (the in-sample error minus the empirical risk, both at \hat{f}) was given by $\omega = \frac{2}{n} \sum_{i=1}^n \text{Cov}(\hat{y}_i, y_i)$
 - the tighter the fit to the training data, the greater the optimism.

Using test data to estimate risk of \hat{f}

- Imagine we find an estimate \hat{f} , and had an infinite test set (separate from the training data).
 - We could use this to find a perfect estimate of R
 - the empirical risk found using the test set converges to $R(\hat{f})$.
- Suppose we have a finite test set
 - the empirical risk found using the test set gives an unbiased estimate of $R(\hat{f})$
 - possible because the training and test set are independent.
- Is this enough?
 - we can estimate the prediction error of an estimator of f
 - but sometimes we would like estimates of R to help us choose the model.

Combatting overfitting III: regularisation

- Choose a flexible class $\bar{\mathcal{F}}$.
- Modify ERM by using a *regularising* term.
- For example

$$\hat{f} = \arg \min_{f \in \bar{\mathcal{F}}} \hat{R}(f) + \lambda \|f\|^2$$

where $\|\cdot\|$ is some norm.

- Called *weight decay* in neural networks.
- We have a free parameter λ , the *regularisation parameter*
 - aim to choose the right trade-off between fit and complexity.
- How choose λ ?
 - usually cross-validation (this lecture!).

Test set → validation set

- We cannot use a test set to help us choose λ , since we need to keep the test set for evaluating our overall approach.
- Instead we use a *validation set*.
- Idea:
 - "hold out" part of our training set - this is called the validation set
 - the validation set is independent from our (now smaller) training set
 - we can estimate the risk by calculating the empirical risk on the validation set.

Problems with using a validation set

- There are potentially two issues with this idea.
 - both are associated with how we split the original training set into a smaller training set and a validation set.
1. If the validation set is small
 - the estimate of the risk will have a high variance.
 2. If the validation set is large
 - the training set will be small
 - the resultant \hat{f} will then be based on only a small data set (compared to the original size n)
 - therefore we do not expect this choice of \hat{f} to be "as good" as if we had trained it on all n points
 - this means that the risk estimated by the validation set will tend to be higher than it would be if we could have trained on all n points.

Leave-one-out cross validation

- *Leave-one-out cross validation* (LOOCV) is designed to address both drawbacks of the validation approach.
- Begin from point 2
 - we do not want the training set to be small, so we choose the validation set to be a single point
 - this would result in estimating the risk using one data point, which will have a very high variance! (point 1).
- Let's split the original training set **many times** into a training and validation set
 - do this for each of the n training points - i.e. each one gets a turn as being the validation set
 - then average the risk estimates obtained for each of these cases.
- Drawback: expensive, since the model is fitted n times.

K -fold cross validation

- K -fold cross validation is the generalisation of LOOCV to the case of using sets of larger than one point as the validation set.
- Define a partition of the n original training points into K pieces
 - let $\kappa : \{1, \dots, n\} \rightarrow \{1, \dots, K\}$ define this partition.
- Use \hat{f}^{-k} to denote the function fitted to the training data using ERM with piece $k \in \{1, \dots, K\}$ removed.
- Cross-validation estimate of $R(\hat{f})$, the risk of the ERM estimate \hat{f} , is given by
$$\hat{R}^{\text{CV}}(\hat{f}) = \frac{1}{n} \sum_{i=1}^n L(Y_i, \hat{f}^{-\kappa(i)}(X_i)).$$
- LOOCV is the case $K = n$.

K -fold cross validation for regularisation

- We may use cross validation for choosing the parameter λ in regularisation.
- Use \hat{f}_λ^{-k} to denote the function fitted to the training data using regularised ERM with parameter λ with piece $k \in \{1, \dots, K\}$ removed.
- Cross-validation estimate of $\hat{R}(\hat{f}_\lambda)$, the risk of the regularised ERM estimate \hat{f}_λ , is given by

$$R^{\text{CV}}(\hat{f}_\lambda) = \frac{1}{n} \sum_{i=1}^n L(Y_i, \hat{f}_\lambda^{-\kappa(i)}(X_i)).$$

- $R^{\text{CV}}(\hat{f}_\lambda)$ may be used as a score for λ
 - typically we might look at a grid of λ values
 - use the best one, or look at the "path" of estimates across different values of λ .

K -fold cross validation for regularisation

- How should we choose K ?
- Large K (recall that we obtain LOOCV when $K = n$) gives:
 - a small bias since the training sets are larger;
 - a large variance since there are few validation points.
- Small K gives:
 - a large bias since the training sets are small and the estimated functions do not perform as they would if trained on more data;
 - a small variance since there are many validation points.
- Often $K = 5$ or $K = 10$ are chosen
 - but the best value will be problem specific.

Bootstrapping

- *Bootstrapping* is an alternative resampling-based technique.
- We will see two uses for it in supervised learning, but it is most commonly used for estimating properties of the sampling distribution of estimators
 - we describe this latter situation first.
- Let $Z = (Z_1, \dots, Z_n)$ be a random sample from some distribution P .
- Let $S(Z)$ be some estimate or statistic calculated from this random sample.
- For some $S(Z)$ (e.g. the sample mean or sample variance) we can analytically calculate various properties of the sampling distribution
 - common example is the variance of the sample mean.
- The bootstrap can help in cases where the analytic calculations are not possible.

Bootstrap: idea

- What is the sampling distribution of $S(Z)$?
 - it's the randomness in S over different samples Z .
- Why might it be difficult to calculate properties of the sampling distribution?
 - these are often integrals over the distribution of $S(Z)$
 - not necessarily analytically tractable.

Bootstrap: idea

- A commonly used idea is to use a *Monte Carlo* approximation to estimate an integral
 - let θ be a random variable with distribution π
 - if $(\theta_1, \dots, \theta_N)$ is an i.i.d. sample from π
 - then

$$\frac{1}{N} \sum_{i=1}^N h(\theta_i)$$

is an unbiased estimate of

$$\mathbb{E}[h(\theta)]$$

where h is some function (which will be the identity if all we need is the expectation, or the squared distance to the mean if we wish to estimate the variance).

- law of large numbers guarantees convergence as $N \rightarrow \infty$.

Bootstrap: idea

- Our case: π is the sampling distribution of $S(Z)$.
- Thus, if we could sample from the sampling distribution of $S(Z)$, we would be able to estimate integrals with respect to this distribution.
- Recall the randomness in $S(Z)$ comes from Z
 - so, if we could sample different random samples Z (each of size n) from P , we would obtain the distribution of $S(Z)$ simply by applying S to these different samples.
- However, it is not obvious how we would sample from this sampling distribution if we only have a single observed random sample.

Bootstrap: idea

- In the bootstrap, the true distribution P is replaced by \tilde{P}_n , the empirical distribution of the random sample X
 - the subscript in \tilde{P}_n denotes that we are using the empirical distribution of n data points.
- This sounds mad, but it works in many cases!
- Recall the plan was to draw a number N of random samples (each of size n) from P .
- Instead we draw a number N of random samples (each of size n) from \tilde{P}_n .
- What does this mean? Let's use an example.

Bootstrap: idea

- Choose $n = 5$ so that $Z = (Z_1, Z_2, Z_3, Z_4, Z_5)$. The following are all samples of size n from \tilde{P}_n
 - $\tilde{Z}^1 = (Z_2, Z_5, Z_2, Z_1, Z_4)$
 - $\tilde{Z}^2 = (Z_4, Z_3, Z_1, Z_5, Z_3)$
 - $\tilde{Z}^3 = (Z_1, Z_2, Z_3, Z_2, Z_2)$
- These are *resamples* of the training data.
- $(S(\tilde{Z}^1), \dots, S(\tilde{Z}^N))$ is then treated as if it is a Monte Carlo sample from the sampling distribution of $S(Z)$
 - e.g. the sample variance of $(S(\tilde{Z}^1), \dots, S(\tilde{Z}^N))$ is the bootstrap estimate of the variance of the sampling distribution of $S(Z)$.

Use of the bootstrap for estimating prediction error

- Generate N resampled bootstrap data sets.
- Let \hat{f}^b denote the function fitted to the b th bootstrap sample.
- First idea:

$$\hat{R}^{\text{boot1}}(\hat{f}) = \frac{1}{N} \frac{1}{n} \sum_{b=1}^N \sum_{i=1}^n L(Y_i, \hat{f}^b(X_i))$$

- we have used different bootstrap sets for training, and the original training sample to validate.
- This will not work, since the bootstrap sets overlap with the set we are using for validation.

Aside: how much overlap?

- What is the probability that the observation is not chosen in any of the n points in the bootstrap sample?

$$\left(1 - \frac{1}{n}\right)^n$$

- What is the probability of observation i being in bootstrap sample b ?

$$1 - \left(1 - \frac{1}{n}\right)^n.$$

- As $n \rightarrow \infty$, this tends to $1 - \exp(-1) \approx 0.632$.

Use of the bootstrap for estimating prediction error

- Next idea:

$$\hat{R}^{\text{boot2}}(\hat{f}) = \frac{1}{n} \sum_{i=1}^n \frac{1}{|C^{-i}|} \sum_{b \in C^{-i}} L(Y_i, \hat{f}^b(X_i))$$

- C^{-i} is the set of bootstrap sets that do not contain point i
- so now the sets on which we are validating do not contain the points on which we trained each model
- note that we need to use enough bootstrap samples such that none of the C^{-i} are empty, or need to omit from the sum the i for which C^{-i} is empty.
- There will still be a bias (an overestimate of the error) for the same reasons as point 2 in cross validation.

Use of the bootstrap for estimating prediction error

- Recall that $\hat{R}(\hat{f})$ underestimates the error.
- To reduce this bias, it turns out that the following estimate is a reasonable idea

$$\hat{R}^{\text{boot3}}(\hat{f}) = 0.368\hat{R}(\hat{f}) + 0.632\hat{R}^{\text{boot2}}(\hat{f}).$$

- The results when using cross validation and the bootstrap are comparable (see p253 in ESL).

Discussion

- We have seen alternative ways of estimating R using resampling
 - cross validation
 - bootstrap.
- We have already seen that the bootstrap has other uses (for estimating properties of sampling distributions).
- Next lecture: "bagging", another use of the bootstrap.

Further reading

- ISLR chapters 5 and 8.
- ESL chapters 8 and 10.
- Bishop chapter 14.

Decision trees, bagging and random forests

ST420 Lecture 23

Richard Everitt

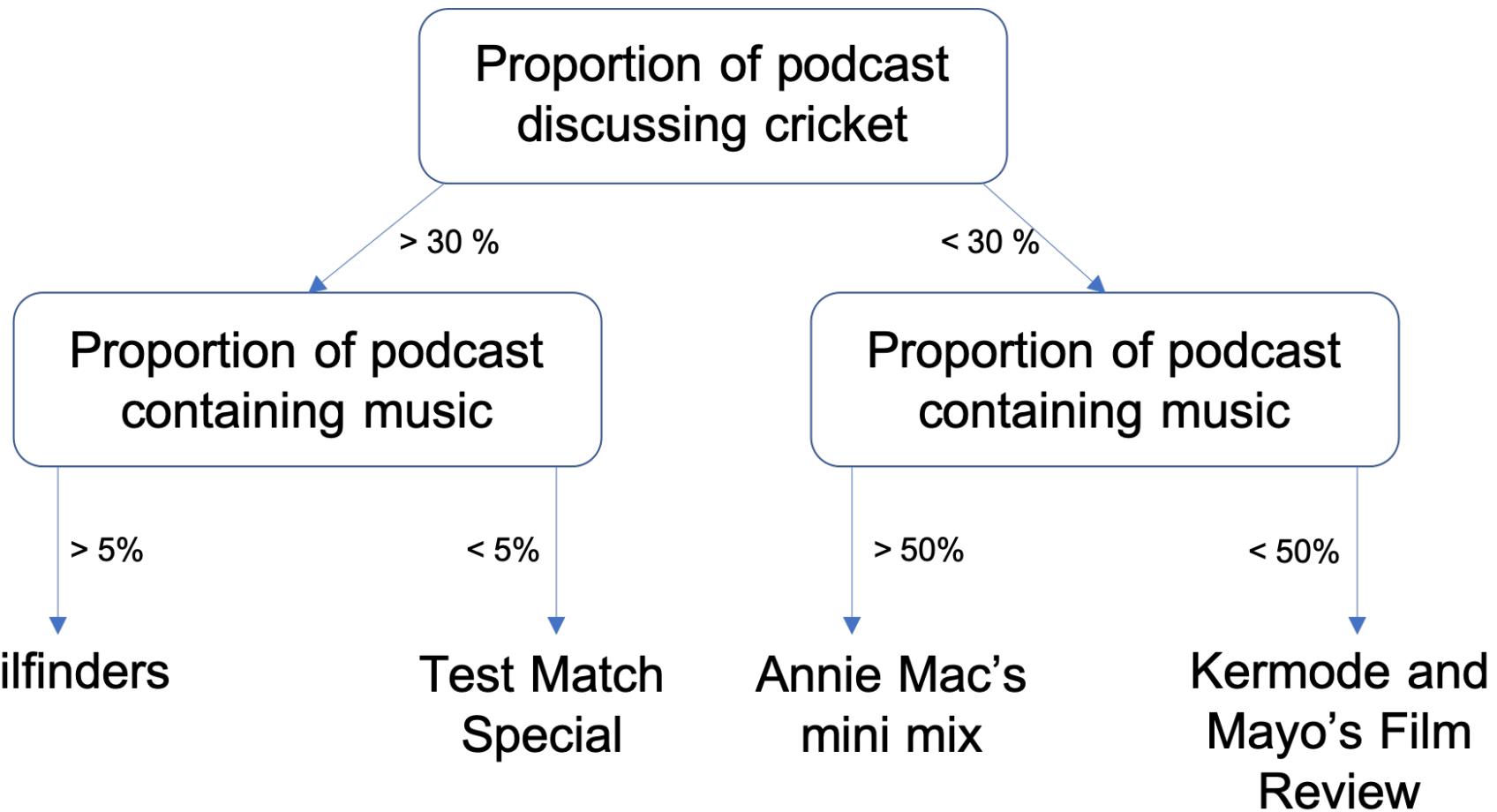
Recap and plan

- Last lecture we introduced two resampling techniques, which we used to estimate the prediction error (risk) of a function estimate
 - cross validation
 - bootstrap.
- The bootstrap also has other uses, one of which we introduce in this lecture
 - "bagging".
- Bagging is particularly useful on tree-based classifiers, so we introduce them first.

Decision trees

- The simplest classifier?
 - classifications made by a sequence of "if-then" rules.
- Used in "expert systems"
 - a branch of AI that was popular in the 1980s
 - many examples in medical diagnosis.
- Plan:
 - give an example of the model;
 - explain more carefully what the model is doing;
 - introduce an approach for estimating the model.

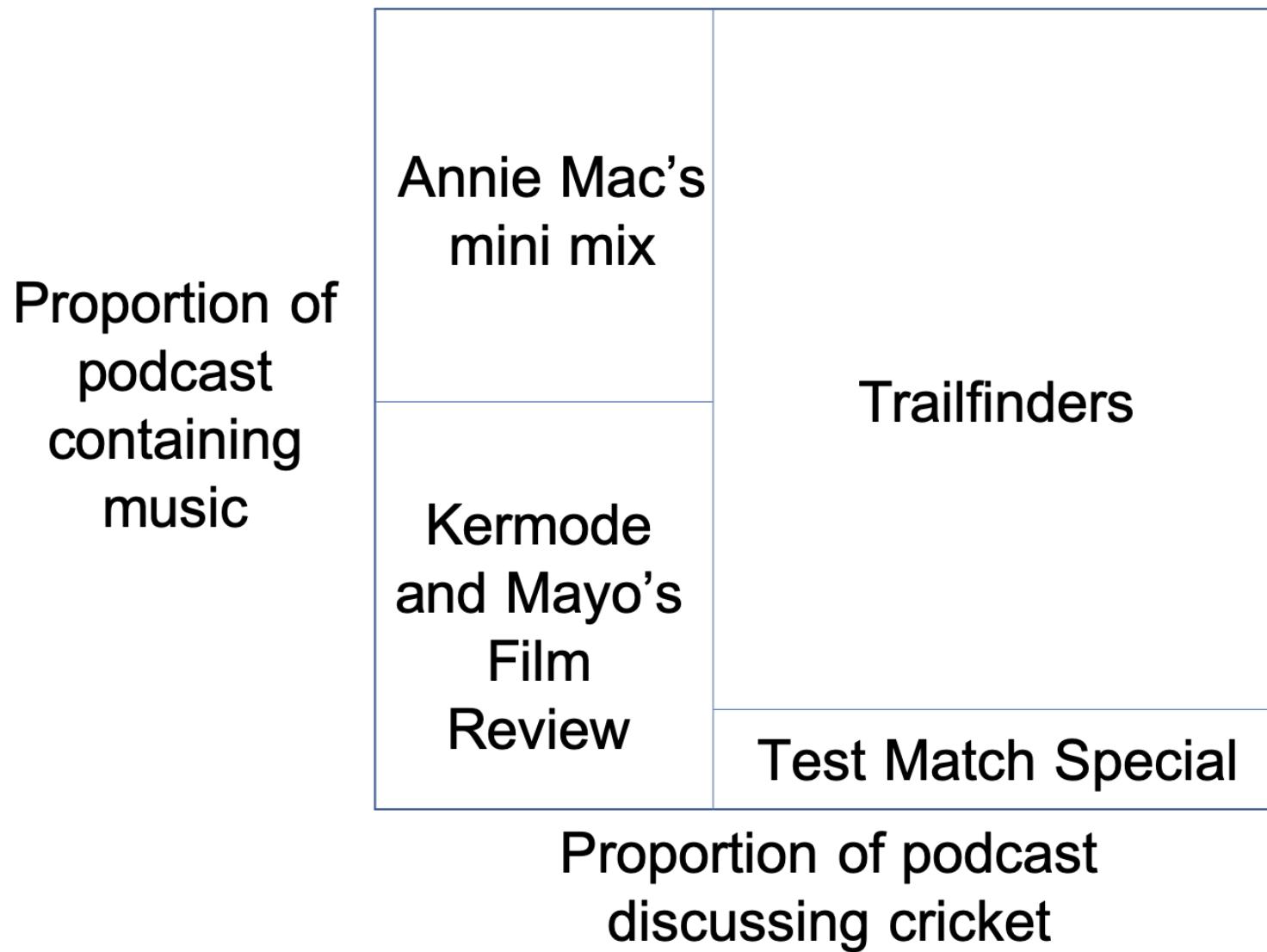
Example: classifying BBC podcasts



Decision trees for regression and classification

- The idea is to partition the space of predictors into different regions
 - the regression/classification is piecewise uniform over this partition.
- The space is partitioned by using thresholds on individual predictors, in a recursive binary fashion
 - therefore the partition is made up of boxes
 - the process can be described using a tree.

Example: classifying BBC podcasts



Classification and regression trees

- Let \mathcal{X} be partitioned into M regions S_1, \dots, S_M .
- $\bar{\mathcal{F}}$ is chosen to consist of functions of the form

$$f(X) = \sum_{m=1}^M c_m I(X \in S_m).$$

- How do we fit an f in this class?
 - what should the c_m be?
 - what should the regions be?

Finding the c_m

- To find c_m in regression
 - minimise squared error loss within each region
 - to do this, take c_m to be the average of the Y_i for which $X_i \in S_m$.
- To find c_m in classification
 - minimise zero-one loss within each region
 - to do this, take c_m to be the most common Y_i for which $X_i \in S_m$.

Finding the partition

- Inferring the optimal partition (for any useful score) is intractable.
- Restricting ourselves to a binary partition still (usually) results in an optimisation problem that is computationally intractable.
- A greedy algorithm is used.
- At the first stage of the method, we will choose a variable to threshold, and choose the threshold. Let the two regions be

$$S_1(j, s) = \{X \mid X^{(j)} \leq s\} \quad S_2(j, s) = \{X \mid X^{(j)} > s\}.$$

Finding the partition

- For regression j and s are chosen to minimise

$$\min_{j,s} \left[\min_{c_1} \sum_{X_i \in S_1(j,s)} (Y_i - c_1)^2 + \min_{c_2} \sum_{X_i \in S_2(j,s)} (Y_i - c_2)^2 \right]$$

where the minimisation over c_1 and c_2 is performed as on the previous slide.

- This process is repeated iteratively, where at the second stage we look to partition the regions found in the first stage, etc.
- This is done until some each region contains some minimum number of training points.

Cost complexity pruning

- A tree created using the method above will result in a large number of elements in the partition
 - we might imagine that we will overfit the model.
- To mitigate overfitting, we prune the tree.
- Consider the score

$$\sum_{m=1}^{|T|} \sum_{i | X_i \in S_m} L(Y_i, f(X_i)) + \lambda |T|,$$

where $|T|$ is the number of terminal nodes in the tree.

- Just as for the lasso, etc, $\lambda = 0$ simply gives us the empirical risk, with (for this case) no penalty on the size of the tree.
- As λ increases, the size of the tree is penalised. At some point as we increase λ , a subtree of the complete tree will obtain a better score than the complete tree.

Cost complexity pruning

- Perform this search iteratively, where at each iteration we look for a subtree of the previous tree, to find a sequence of subtrees.
- Then use cross validation to choose λ .
- Output the tree that corresponds to this choice of λ .

Performance

- The full tree (without pruning) gives a very flexible classifier
 - low bias
 - usually very high variance.
- Pruning introduces bias and reduces variance.
- However, predictive performance is often poor due to overfitting
 - the tree is very sensitive to the choice of training data.
- "Bagging" will help us with this.

Bagging

- *Bagging* is "bootstrap aggregating".
- The "bootstrap" part is that we resample the training data N times, to produce bootstrapped data sets $\mathcal{T}^1, \dots, \mathcal{T}^N$.
- We then fit a decision tree to each bootstrapped training data using the procedure described on the previous slides.

Bagging

- Bagging then "aggregates" these trees. If \hat{f}^b is the function fit for bootstrapped training set b , then our prediction when using bagging is
 - for regression

$$\hat{f}^{\text{bag}}(X) = \frac{1}{N} \sum_{b=1}^N \hat{f}^b(X);$$

- for classification

$$\hat{f}^{\text{bag}}(X) = \text{mode}\left\{\hat{f}^b(X)\right\}.$$

Properties

- Suppose that the random samples we train the individual trees on were independently drawn from P (rather than from the empirical distribution).
- For a fixed x , suppose the $\hat{f}^b(x)$ are independent over b .
- In the regression context, for a fixed x , we have

$$\begin{aligned}\mathbb{V}\left[\hat{f}^{\text{bag}}(x)\right] &= \mathbb{V}\left[\frac{1}{N} \sum_{b=1}^N \hat{f}^b(x)\right] \\ &= \frac{1}{N^2} \sum_{b=1}^N \mathbb{V}\left[\hat{f}^b(x)\right] \\ &= \frac{1}{N} \mathbb{V}\left[\hat{f}(x)\right].\end{aligned}$$

- So, in an idealised case, we hope that by bagging we reduce the variance, without introducing additional bias.

Properties

- VC dimension:
 - when using p predictors, each binary rule has a VC dimension of $p + 1$
 - for a tree that uses k sets in the partition, the VC dimension is $\leq (p + 1)k$.
- When using bagging, the risk may be estimated by using the points that are not in each bootstrap sample
 - see the bootstrap estimators of risk introduced in the last lecture
 - in the context of bagging, the points that are not in the bootstrap sample are known as the "out-of-bag" points.
- Bagging is useful for classifiers that overfit
 - e.g. it can be useful using this approach for neural networks.
- Bagging results in us losing some interpretability of a classifier.

Limitation of bagging for decision trees

- Recall that in the ideal independent case we can reduce the variance by a factor of N when averaging over N individual classifiers.
- When the variables are identically distributed with variance σ^2 , but have positive pairwise correlation of ρ , the variance of the average becomes

$$\rho\sigma^2 + \frac{1 - \rho}{N} \sigma^2.$$

- When using bagging on decision trees, we often find that the predictions of trees \hat{f}^b inferred for different bootstrap samples are often highly correlated
 - suppose there is one predictor that is very important in affecting the risk
 - then all of the bagged trees tend to choose to threshold this variable first, with a similar choice of threshold
 - these trees end up looking very similar, and the resulting correlation limits the benefit of bagging.

Improved bagging for decision trees: random forests

- A *random forest* is simply a classification or regression tree with bagging, each tree being fitted using the procedure earlier in the lecture
 - except for the difference that before each binary split is made, a random selection of m of the p predictor variables is made available to split.
- This restriction results in the individual trees being more dissimilar
 - results in more variance reduction.
- Random forests can exhibit very good performance and are a very popular machine learning technique.
- Next time we look at an alternative approach: "boosting".

Further reading

- ISLR chapters 5 and 8.
- ESL chapters 8 and 10.
- PRML chapter 14.

Boosting

ST420 Lecture 24

Richard Everitt

Recap

- We started looking at resampling-based methods for estimating the excess risk.
- We looked at bagging: combining high-variance estimators of f trained on bootstrapped datasets, as a way to achieve a lower variance estimator.
- There is an alternative to bagging, called "boosting"
 - to introduce it, we first go back to an idea from PAC learning.

PAC learnability

- Under the assumptions of performing binary classification, using zero-one loss, that $R(\bar{f}^*) = 0$, and also assuming $\mathcal{X} = [0, 1]^p$...
- We say that an algorithm *PAC learns* \mathcal{F} using $\bar{\mathcal{F}}$ if, for any true $f \in \mathcal{F}$, and for any $0 < \varepsilon, \delta < 1$, the algorithm:
- with probability at least $1 - \delta$, outputs a hypothesis $\hat{f} \in \bar{\mathcal{F}}$ with $R(\hat{f}) \leq \varepsilon$;
- runs in polynomial time in $1/\varepsilon, 1/\delta, \text{size}(f)$ and p .
- Here \hat{f} is simply some chosen member of $\bar{\mathcal{F}}$, not necessarily chosen using ERM.
- We will not define $\text{size}(f)$ here
 - it is essentially the complexity of representing f .

Strong vs weak PAC learnability

- The above is sometimes known as a definition of *strong* PAC learning
 - to be satisfied, the algorithm needs to be able to achieve an arbitrarily small ε and δ .
- *Weak* PAC learning uses a fixed ε and δ .
- We say that an algorithm is a *weak PAC learning* algorithm if there exist fixed polynomials $p_\delta(\cdot, \cdot)$ and $p_\varepsilon(\cdot, \cdot)$ such that the algorithm outputs \hat{f} such that

$$R(\hat{f}) \leq \frac{1}{2} - \frac{1}{p_\varepsilon(p, \text{size}(f))}$$

with probability at least

$$\frac{1}{p_\delta(p, \text{size}(f))}.$$

Strong vs weak PAC learnability

- The definition of weak PAC learning is very weak
 - the algorithm only needs to be slightly better than a random guess!
- However:
 - a problem can be weak-learned if and only if it can be strong-learned!
- This insight is used in *boosting*, in which a weak PAC learning algorithm can be converted into a strong PAC learning algorithm
 - by boosting confidence δ
 - and accuracy ϵ .
- We will describe this approach in this lecture.

Bagging improves confidence

- Recall that bagging averages multiple classifiers
 - suppose that each is a weak learner
 - the aim is to reduce variance through the averaging.
- "Confidence" refers to a bound on the probability of the risk being bigger than some ϵ
 - it is bounding the probability of being in the tail
 - as the variance decreases, this bound can decrease
 - recall Chebyshev's inequality that most directly makes this link.

Improving accuracy?

- Improving confidence requires reducing variance, which we can achieve through bagging.
- A classification/regression technique is more accurate if the size of the difference between the risk of a fitted function and the optimal risk
 - to improve accuracy we must reduce bias
 - reducing bias is more difficult than reducing variance.
- We first look at an idea for reducing the errors of weak learners by combining them
 - this introduces the concepts of *boosting*.
- We then introduce and analyse an approach that is widely applicable in practice: "AdaBoost".

Sketch of original idea

- A binary classification problem can be weak-learned if and only if it can be strong-learned
 - proof in Schapire, Robert E. (1990). "The Strength of Weak Learnability". *Machine Learning*. 5 (2): 197–227
 - points the way to the idea of boosting.
- We will only give a sketch here.
- The idea is to train three weak classifiers, and to combine them to do the classification
 - the error of the combination is guaranteed to be less than the error of each weak learner
 - in contrast to bagging, the training of each weak classifier is done sequentially
 - each weak classifier is trained based on the test results of the previous classifier.

Sketch of original idea

1. Divide our training set \mathcal{T} into three pieces: $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$.
2. Train a weak classifier on \mathcal{T}_1 , obtaining \hat{f}^1 .
3. Test \hat{f}^1 on \mathcal{T}_2 , i.e. classify all of the points in \mathcal{T}_2 using \hat{f}^1 .
4. Train a weak classifier on $\mathcal{T}'_2 \subseteq \mathcal{T}_2$, where \mathcal{T}'_2 consists of an equal number of correct and wrongly classified points from \mathcal{T}_2 . The classifier trained on this set is denoted by \hat{f}^2 .
5. Test both \hat{f}^1 and \hat{f}^2 on \mathcal{T}_3 , i.e. classify all of the points in \mathcal{T}_3 using both \hat{f}^1 and \hat{f}^2 .
6. Train a weak classifier on $\mathcal{T}'_3 \subseteq \mathcal{T}_3$, where \mathcal{T}'_3 consists of the points where \hat{f}^1 and \hat{f}^2 disagree. The classifier trained on this set is denoted by \hat{f}^3 .

Sketch of original idea

- To classify a new point we
 - classify it using \hat{f}^1, \hat{f}^2 , and if they agree, taking this as the result
 - if they disagree, classify the point using \hat{f}^3 , and use this as the result.
- This is equivalent to using all three of \hat{f}^1, \hat{f}^2 and \hat{f}^3 and choosing the most commonly voted for class.
- If the error (proportion of misclassifications) of each weak classifier is ε , which we know is < 0.5 due to it being a weak learner, the error of the combination is $3\varepsilon^2 - 2\varepsilon^3$
 - $3\varepsilon^2 - 2\varepsilon^3 < \varepsilon$ for $0 < \varepsilon < 0.5$.
- The key idea was to train each classifier on points misclassified by the previous classifiers
 - this reduces the bias
 - this is the key idea of *boosting*.

Sketch of classification of a new point



AdaBoost

- Following the early work, which was motivated by theory, boosting was turned into a general and widely applicable algorithm with the advent of *AdaBoost* (adaptive boosting).
- Idea: use **adaptively weighted** training data, beginning the method with equally weighted points $w_i^{(1)} = 1/n$ for all $1 \leq i \leq n$.
- Algorithm (step j):
 - fit a weak classifier \hat{f}^j to the weighted training data using weighted empirical risk
$$\sum_{i=1}^n w_i^{(j)} L(Y_i, f(X_i));$$
 - calculate the contribution β^j that classifier j will make to the final classifier;
 - find the points that the classifier misclassified, and give them more weight - update the weights to give $w_i^{(j+1)}$ for all $1 \leq i \leq n$;
 - go to the first step.

AdaBoost

- The final classifier is given by a weighted combination, using weights, β^j of the \hat{f}^j
 - for regression, we use

$$\hat{f}(X) = \sum_{j=1}^m \beta^j \hat{f}^j(X)$$

- for binary classification with response in $\{-1, 1\}$, we use

$$\hat{f}(X) = \text{sign}\left(\sum_{j=1}^m \beta^j \hat{f}^j(X)\right).$$

- As previously, we see that the idea is to sequentially train the weak classifiers, such that each classifiers focuses more attention at points that were classified poorly by previous classifiers.
- The original algorithm gives particular expressions for β^j and $w_i^{(j+1)}$
 - we will show that these can be derived though a loss function perspective.

AdaBoost: loss function perspective

- Using

$$\hat{f}(X) = \sum_{j=1}^m \beta^j \hat{f}^j(X),$$

we see that boosting can be seen as using the individual classifiers \hat{f}^j as features.

- A distinguishing feature of boosting is that the training of the classifiers \hat{f}^j and the estimation of the coefficients β^j is performed sequentially.

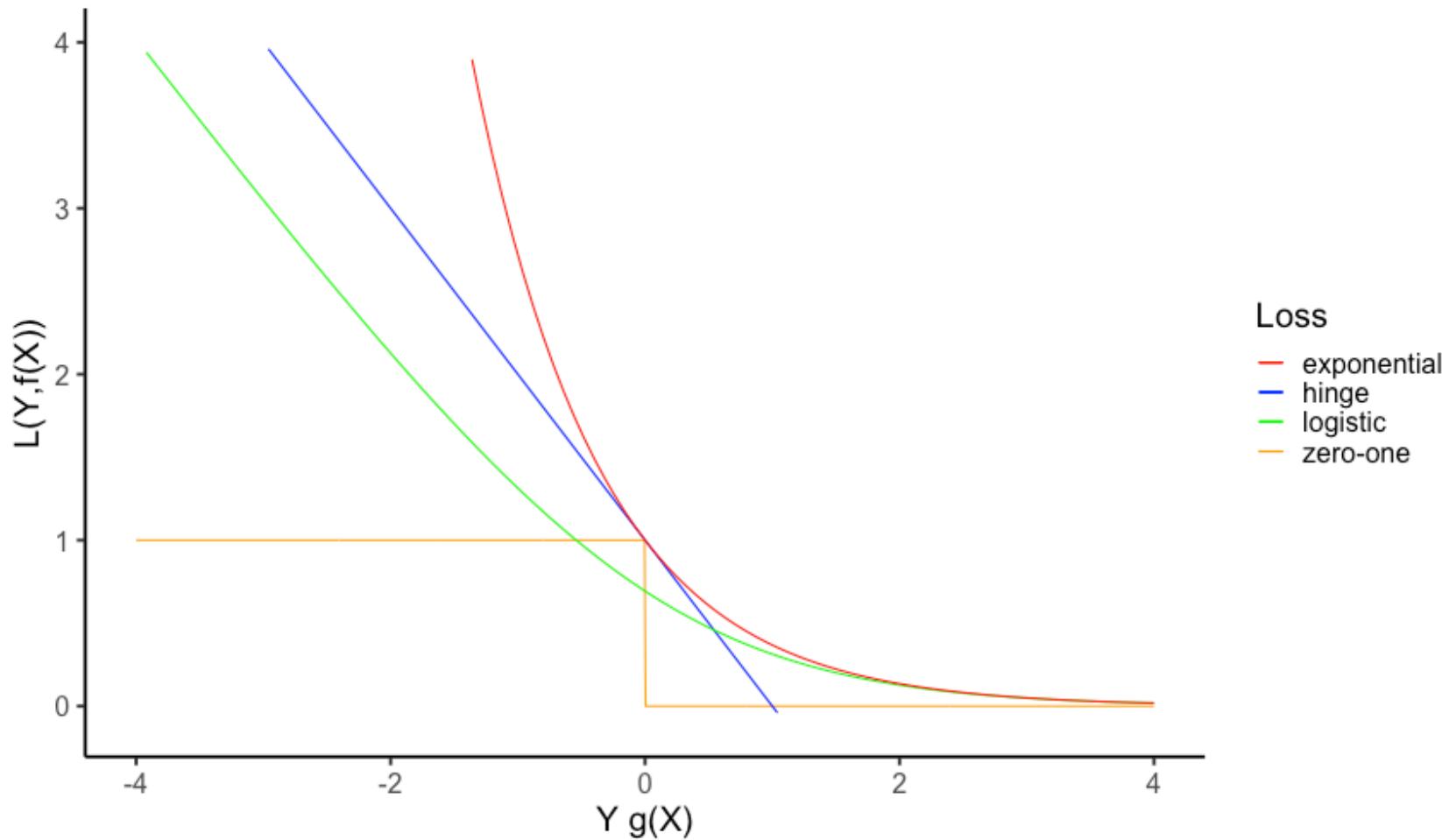
AdaBoost: loss function perspective

- At step j of AdaBoost the fit of \hat{f}^j and estimation of β^j is performed on the weighted training data by minimising *exponential loss*, with all previous classifiers and coefficients being fixed.
- Exponential loss

$$L(Y, f(X)) = \exp(-Yg(X)),$$

where g is the function we will take the sign of in order to perform binary classification.

Sketch of loss functions for classification



AdaBoost: loss function perspective

- At iteration j of AdaBoost, we wish to find

$$\arg \min_{\beta^j, f^j} \sum_{i=1}^n \exp \left(-Y_i \left(\sum_{j'=1}^{j-1} \beta^{j'} \hat{f}^{j'}(X_i) + \beta^j f^j(X_i) \right) \right).$$

- Let $w_i^j = \exp \left(-Y_i \sum_{j'=1}^{j-1} \beta^{j'} \hat{f}^{j'}(X_i) \right)$. Then we can:

- find

$$\arg \min_{\beta^j, f^j} \sum_{i=1}^n w_i^j \exp(-\beta^j f^j(X_i));$$

- easily find w_i^{j+1} from w_i^j and $\beta^j f^j(X_i)$;
- view w_i^j as a weight applied to each observation.

AdaBoost: loss function perspective

- We wish to find

$$\arg \min_{\beta^j, f^j} \sum_{i=1}^n w_i^j \exp(\beta^j Y_i f^j(X_i)).$$

- First, for any $\beta^j > 0$, the solution for f^j is

$$\arg \min_{f^j} \sum_{i=1}^n w_i^j I(Y_i \neq f^j(X_i)).$$

- Then, the solution for β^j is

$$\beta^j = \frac{1}{2} \log \frac{1 - \text{err}^j}{\text{err}^j},$$

where err^j is the minimised error rate

$$\text{err}^j = \frac{\sum_{i=1}^n w_i^j I(Y_i \neq \hat{f}^j(X_i))}{\sum_{i=1}^n w_i^j}.$$

Properties

- Using other loss functions is also possible, but not all choices are as computationally convenient.
- Boosting is predominantly a bias reduction technique, so often the weak classifiers are chosen from a very restrictive (i.e. high bias, low variance) family
 - when used on decision trees, a "stump" (one binary split) is often used.
- The VC dimension of a boosting combination of m weak classifiers, each of VC dimension d , is

$$O(m(d+1)(3\log(m(d+1))+2)).$$

Review

- In the past two lectures we have described the two most popular methods for combining classifiers
 - bagging focuses on reducing variance, hence improving confidence
 - boosting focuses on reducing bias, hence improving accuracy.
- In the next lecture we discuss a method for performing bootstrapping in the big data case.

Further reading

- ISLR chapters 5 and 8.
- ESL chapters 8 and 10.
- PRML chapter 14.

The bag of little bootstraps

ST420 Lecture 25

Richard Everitt

Recap

- Big data topics covered in this module
 - computational (mentioned earlier, and today's focus)
 - algorithmic (to come at the end of the module)
 - statistical (mostly discussed earlier).
- Today
 - how to use bootstrapping when n is large.

Bootstrapping

- *Bootstrapping* is a resampling-based technique.
- Commonly used for estimating properties of the sampling distribution of estimators.
- Let $Z = (Z_1, \dots, Z_n)$ be a random sample from some distribution P .
- Let $S(Z)$ be some estimate or statistic calculated from this random sample.
- For some $S(Z)$ (e.g. the sample mean or sample variance) we can analytically calculate various properties of the sampling distribution
 - common example is the variance of the sample mean.
- The bootstrap can help in cases where the analytic calculations are not possible.

Bootstrap: idea

- What is the sampling distribution of $S(Z)$?
 - it's the randomness in S over different samples Z .
- Why might it be difficult to calculate properties of the sampling distribution?
 - these are often integrals over the distribution of $S(Z)$
 - not necessarily analytically tractable.

Bootstrap: idea

- A commonly used idea is to use a *Monte Carlo* approximation to estimate an integral
 - let θ be a random variable with distribution π
 - if $(\theta^1, \dots, \theta^N)$ is an i.i.d. sample from π
 - then

$$\frac{1}{N} \sum_{i=1}^N h(\theta^i)$$

is an unbiased estimate of

$$\mathbb{E}[h(\theta)]$$

where h is some function (which will be the identity if all we need is the expectation, or the squared distance to the mean if we wish to estimate the variance).

- law of large numbers guarantees convergence as $N \rightarrow \infty$.

Bootstrap: idea

- Our case: π is the sampling distribution of $S(Z)$.
- Thus, if we could sample from the sampling distribution of $S(Z)$, we would be able to estimate integrals with respect to this distribution.
- Recall the randomness in $S(Z)$ comes from Z
 - so, if we could sample different random samples Z (each of size n) from P , we would obtain the distribution of $S(Z)$ simply by applying S to these different samples.
- However, it is not obvious how we would sample from this sampling distribution if we only have a single observed random sample.

Bootstrap: idea

- In the bootstrap, the true distribution P is replaced by \tilde{P}_n , the empirical distribution of the random sample Z
 - the subscript in \tilde{P}_n denotes that we are using the empirical distribution of n data points.
- This sounds mad, but it works in many cases!
- Recall the plan was to draw a number N of random samples (each of size n) from P .
- Instead we draw a number N of random samples (each of size n) from \tilde{P}_n .
- What does this mean? Let's use an example.

Bootstrap: idea

- Choose $n = 5$ so that $Z = (Z_1, Z_2, Z_3, Z_4, Z_5)$. The following are all samples of size n from \tilde{P}_n ; *resamples* of the training data
 - $\tilde{Z}^1 = (Z_2, Z_5, Z_2, Z_1, Z_4)$
 - $\tilde{Z}^2 = (Z_4, Z_3, Z_1, Z_5, Z_3)$
 - $\tilde{Z}^3 = (Z_1, Z_2, Z_3, Z_2, Z_2)$
 - ...
- $(S(\tilde{Z}^1), \dots, S(\tilde{Z}^N))$ is then treated as if it is a Monte Carlo sample from the sampling distribution of $S(Z)$
 - e.g. the sample variance of $(S(\tilde{Z}^1), \dots, S(\tilde{Z}^N))$ is the bootstrap estimate of the variance of the sampling distribution of $S(Z)$.
- Let ω denote the property of interest of the sampling distribution.

Bootstrapping with big data

- Problems...
- What if n is so large that:
 - the computational cost of repeated resamples of size n is too large?
 - our sample $Z = (Z_1, \dots, Z_n)$ cannot be stored in RAM, or even on one computer, let alone multiple samples of size n ?
- We would like there to exist a bootstrapping procedure that does not suffer from these drawbacks.
- Kleiner, A., Talwalkar, A., Sarkar, P., and Jordan, M. I. (2014). A scalable bootstrap for massive data. *Journal of the Royal Statistical Society Series B* 76(4), 795–816.

First idea: use a subsample

- Subsample $Z = (Z_1, \dots, Z_n)$ and work instead on this subsample.
- Let $\zeta = (\zeta_1, \dots, \zeta_\eta)$ be a random subset of Z of size $\eta < n$
 - i.e. ζ is obtained by sampling uniformly without replacement from Z .
- η should be chosen to be small enough such that if we work only on a sample of this size, we do not run into computational time or memory problems.
- We are going to use a (slightly modified) bootstrap on this subsample.

Second idea: little bootstrap

- Here we consider what it looks like to use a bootstrap on the subsample **without** any modification.
- Choose $\eta = 3$ and suppose that $\zeta = (Z_1, Z_2, Z_3)$. The following are all resamples of size η from the empirical distribution of ζ
 - $\tilde{\zeta}^1 = (Z_2, Z_3, Z_2)$
 - $\tilde{\zeta}^2 = (Z_2, Z_3, Z_1)$
 - $\tilde{\zeta}^3 = (Z_1, Z_2, Z_1)$
 - ...
- These are *resamples* of ζ .

Second idea: little bootstrap

- Suppose we now apply S to each resample.
- Find $(S(\tilde{\zeta}^1), \dots, S(\tilde{\zeta}^N))$
- Can this be treated as if it is a Monte Carlo sample from the sampling distribution of $S(Z)$?
- What happens if we take, for example, the sample variance of $(S(\tilde{\zeta}^1), \dots, S(\tilde{\zeta}^N))$?
 - is this a good estimate of the variance of the sampling distribution of $S(Z)$?
- No!
 - the sampling distribution of $S(Z)$ is quite different to the distribution of $(S(\tilde{\zeta}^1), \dots, S(\tilde{\zeta}^N))$ since the former is based on a sample of size n and the latter on resamples of size η .

Second idea: little bootstrap

- Solution: modify the process so that although ζ is of size η , we make $\tilde{\zeta}^b$ of size n for each bootstrap resample
 - to do this, compose each resample by sampling with replacement n times from ζ .
- For example, suppose that $\zeta = (\zeta_1, \zeta_2, \zeta_3)$. The following are all resamples of size n from the empirical distribution of ζ
 - $\tilde{\zeta}^1 = (\zeta_2, \zeta_3, \zeta_2, \zeta_2, \zeta_1)$
 - $\tilde{\zeta}^2 = (\zeta_2, \zeta_3, \zeta_1, \zeta_3, \zeta_1)$
 - $\tilde{\zeta}^3 = (\zeta_1, \zeta_2, \zeta_1, \zeta_3, \zeta_2)$
 - ...

Second idea: little bootstrap

- $\left(S\left(\tilde{\zeta}^1\right), \dots, S\left(\tilde{\zeta}^N\right)\right)$ is then treated as if it is a Monte Carlo sample from the sampling distribution of $S(Z)$
 - e.g. the sample variance of $\left(S\left(\tilde{\zeta}^1\right), \dots, S\left(\tilde{\zeta}^N\right)\right)$ is the bootstrap estimate of the variance of the sampling distribution of $S(Z)$.

Second idea: little bootstrap

- Problem: these resamples are of size n , which is what we were trying to avoid!
- Solution: notice that to store the $\tilde{\zeta}^b$, we do not need to store n points
 - for each resample, we only need to store η numbers $c^b = (c_1, \dots, c_\eta)$
 - c_i is the number of times the i th element of the ζ vector was chosen in the resample.
- The resamples on the previous slide can be represented as
 - for $b = 1, c^1 = (1, 3, 1)$
 - for $b = 2, c^2 = (2, 1, 2)$
 - for $b = 3, c^3 = (2, 2, 1)$
 - ...
- Note that in this form, we can see the indices of each resample as being a sample of size n from a multinomial distribution with η outcomes.

Second idea: little bootstrap

- For each $\tilde{\zeta}^b$, $S(\tilde{\zeta}^b)$ needs to be calculated using only ζ and the sample c^b .
- This is usually simple. e.g. where S is a sum, we have

$$S(\tilde{\zeta}^b) = \sum_{i=1}^n c_i^b \zeta_i.$$

- We then use $(S(\tilde{\zeta}^1), \dots, S(\tilde{\zeta}^N))$ to estimate ω .

Third idea: bagging

- Problem: although this "little" bootstrap gives an estimate of what we want (ω), it will have a high variance
 - it will be highly dependent on the subsample.
- Solution: use bagging to average over multiple estimates of ω , found from different subsamples.
- Rather than choosing one subsets of size η , choose s subsets of size η .

Bag of little bootstraps

- For $j = 1 : s$
 - Sample $\zeta(j)$, a subset of size η , without replacement from Z
 - Find estimate $\hat{\omega}(j)$ using the bootstrap procedure described on the previous slides.
- EndFor
- Final estimate

$$\hat{\omega} = \frac{1}{s} \sum_{j=1}^s \hat{\omega}(j).$$

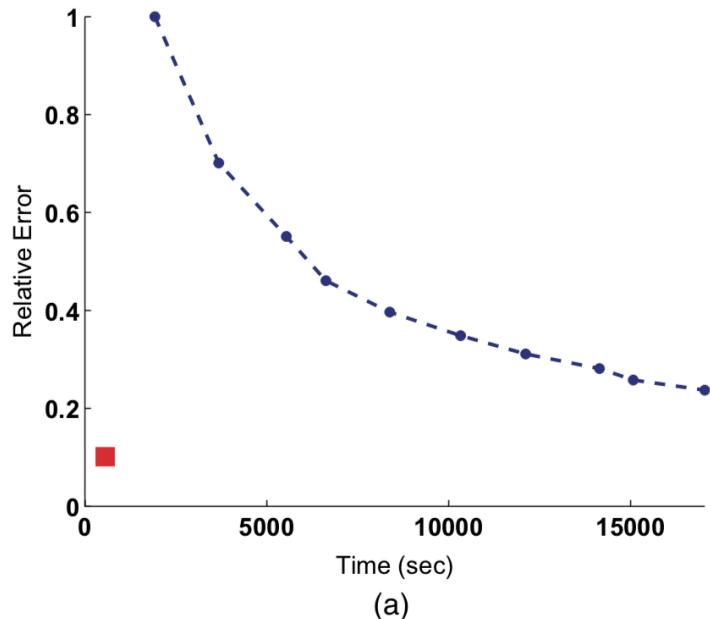
Theoretical properties

- As $\eta, n \rightarrow \infty$, for fixed s , $\hat{\omega} \rightarrow \omega$ in probability.
- The only assumption required on the subsample size η is that it $\rightarrow \infty$ with n
 - there is no assumption needed about the rate at which this happens
 - so η only needs to grow very slowly with n .
- For the standard bootstrap, the rate at which this convergence happens is $O(1/n)$
 - i.e. $|\hat{\omega} - \omega|$ is of this order.
- For the bag of little bootstraps, the same rate holds, as long as s and η are sufficiently large
 - this still holds if $\eta/n \rightarrow 0$.

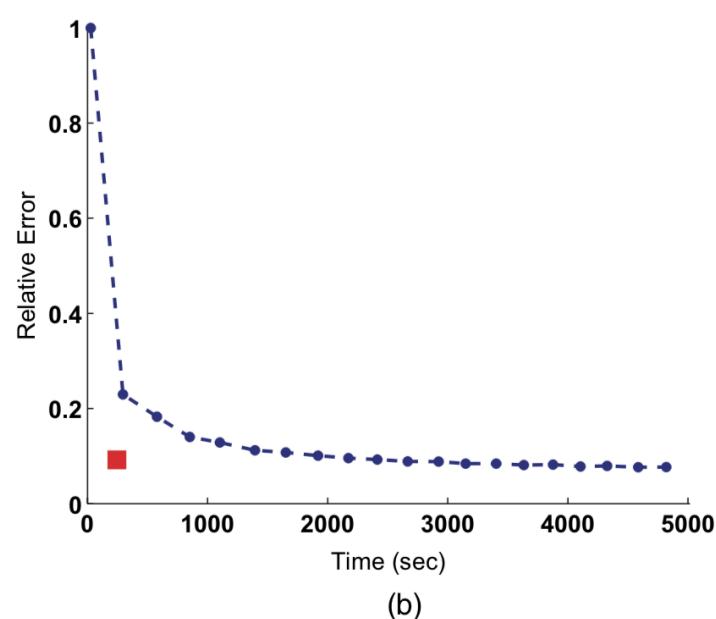
Performance

- From Kleiner et al. (2014)

Scalable Bootstrap 809



(a)



(b)

Fig. 4. Relative error *versus* processing time for the BLB with $b = n^{0.7}$ (■) and the bootstrap (●) on 150 Gbytes of data in the classification setting (because the BLB's computation is fully parallelized across all subsamples, we show only the processing time and relative error of the BLB's final output): (a) results with the full data set stored only on disc; (b) results with the full data set cached in memory

Summary

- This is an example of where statistical ideas have been used to avoid a computational problem.
- From hereon we focus on the curse of dimensionality in p
 - fundamental issue
 - impact on non-parametric methods
 - impact on MCMC ("algorithmic").

The curse of dimensionality

ST420 Lecture 26

Richard Everitt

Recap

- Big data topics covered in this module
 - computational
 - algorithmic (to come at the end of the module)
 - statistical.
- Today
 - we have already looked at linear regression when p is large
 - can we use non-parametric approaches?

k -nearest neighbour

- Recall the k -nearest neighbour estimator for regression

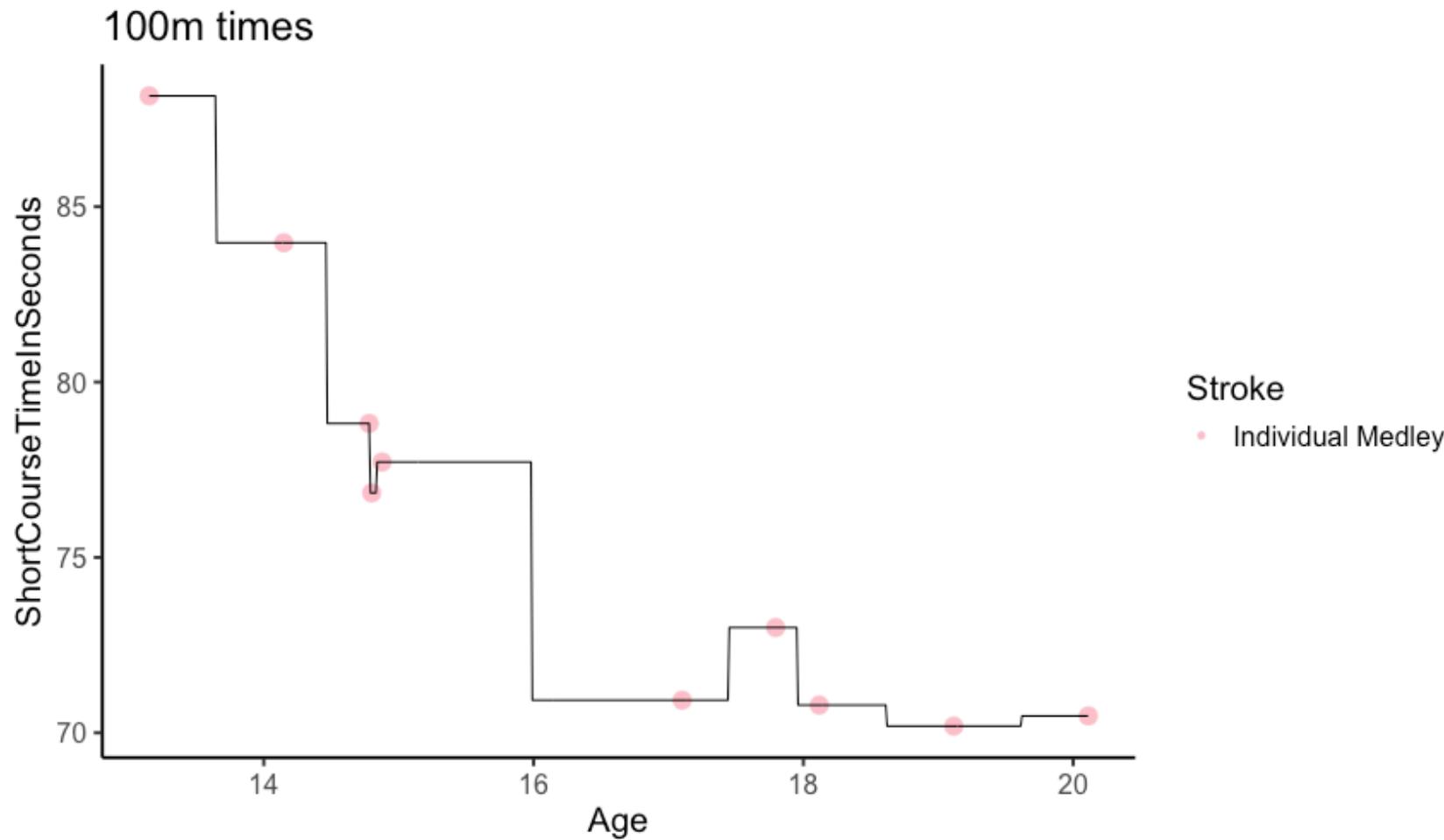
$$\hat{Y} = \frac{1}{k} \sum_{\{l|x_{(l)} \in N_k(x)\}} y_{(l)}$$

where $N_k(x)$ is the neighbourhood of x defined by the k closest (according to some metric) points in the training data. Also recall

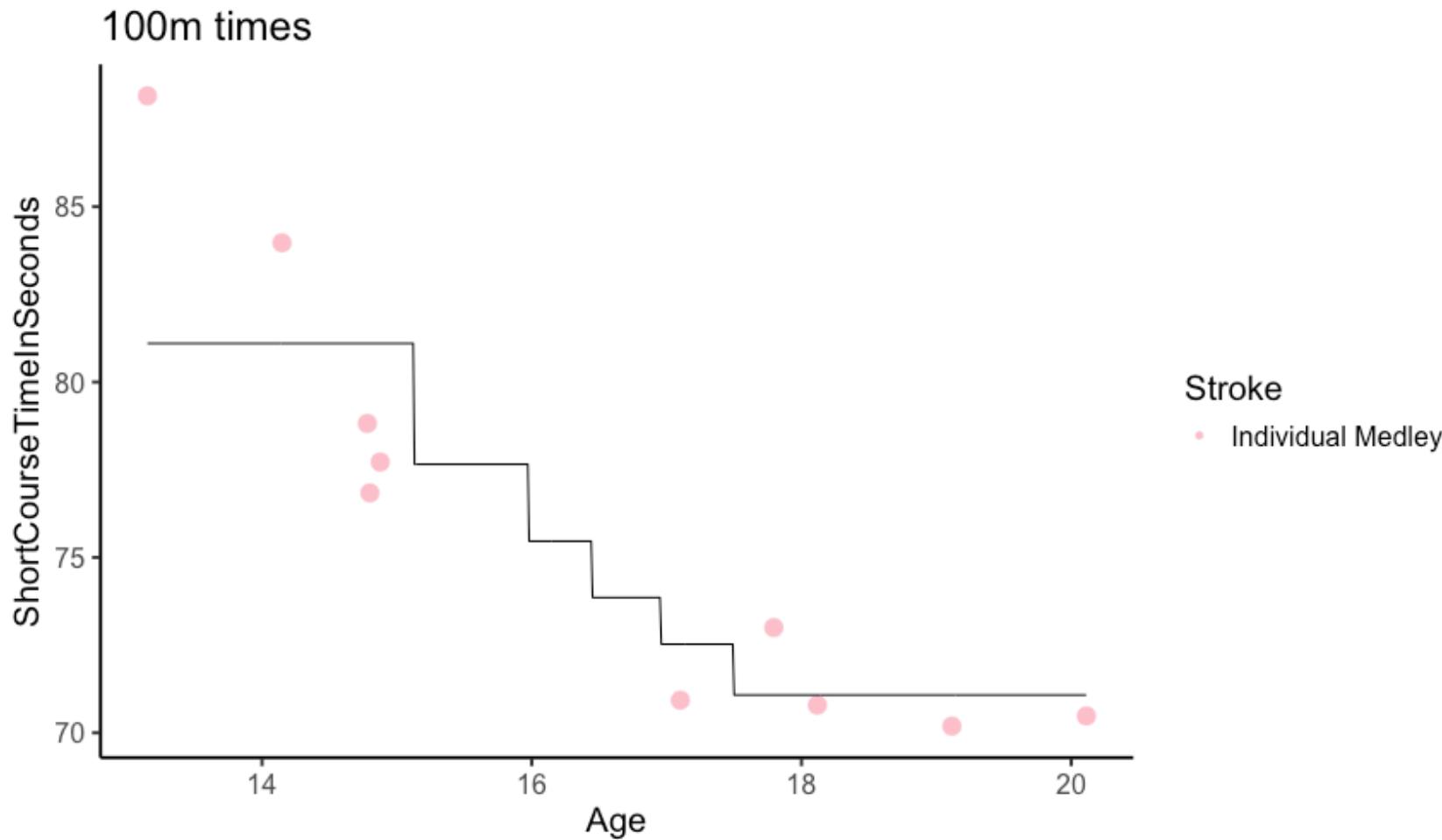
$$\mathbb{E}_Y \mathbb{E}_{\mathcal{T}} \left[L(Y, \hat{f}(x_0)) \mid x_0 \right] = \sigma_\epsilon^2 + \left[f(x_0) - \frac{1}{k} \sum_{l=1}^k f(x_{(l)}) \right]^2 + \sigma_\epsilon^2/k.$$

- σ_ϵ^2 is the irreducible error
- $f(x_0) - \frac{1}{k} \sum_{l=1}^k f(x_{(l)})$ is the bias, which increases with k (since we are including $x_{(l)}$ far away from x_0)
- σ_ϵ^2/k is the variance (recall this is the expression for the variance of a sample mean), which reduces as k increases.

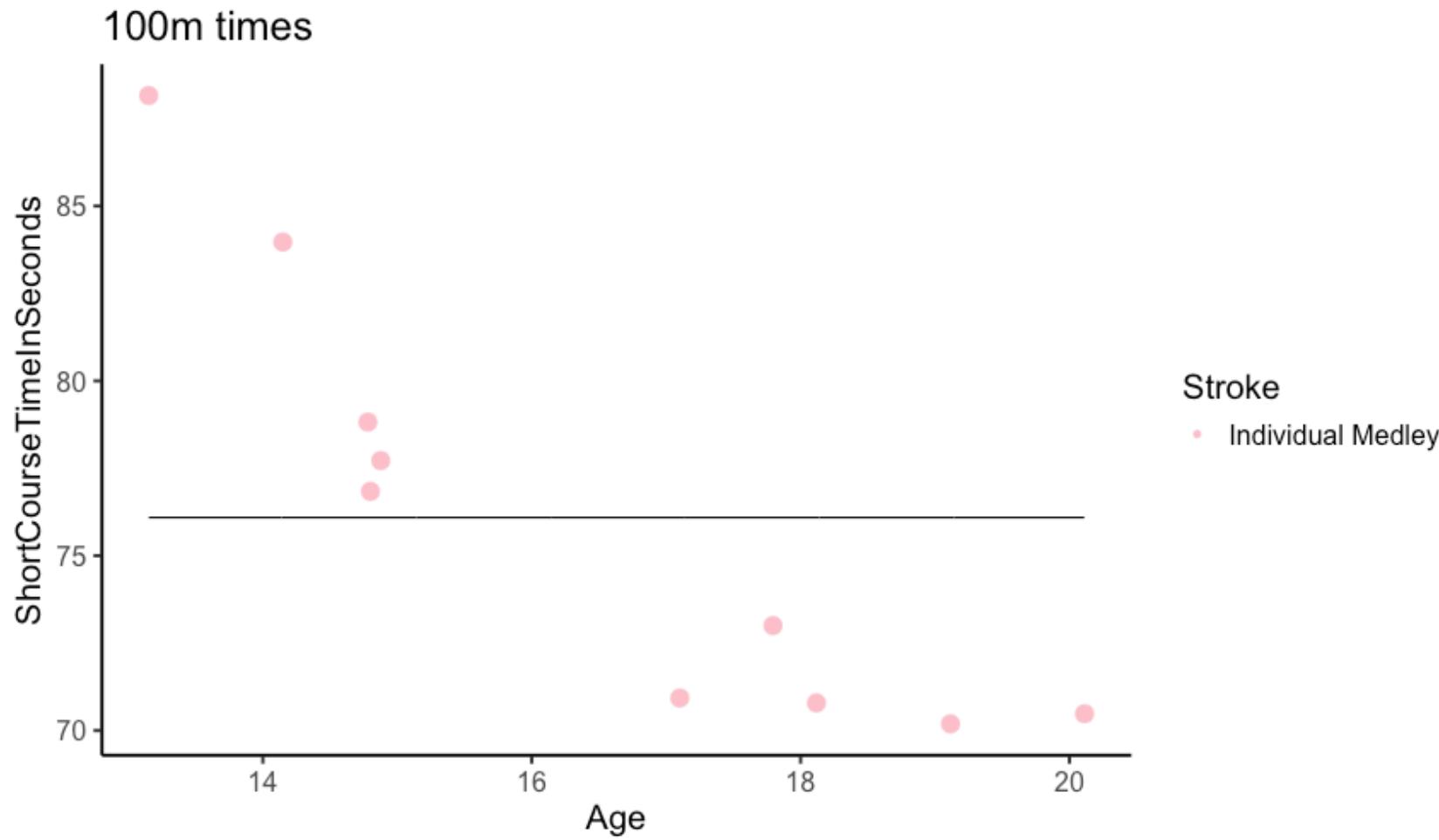
1-nearest neighbour regression



5-nearest neighbour regression



10-nearest neighbour regression



k -nearest neighbour

- How does the dimension p affect the bias and variance?
- The key consideration is how to choose k
 - we need it large enough to give smooth estimates so that the variance is not high
 - we cannot make it too large, or the estimates will not be local.
- One idea:
 - use 0.01 of the total number of data points, i.e. $k = 0.01n$.
- When using this strategy, what happens as p increases?

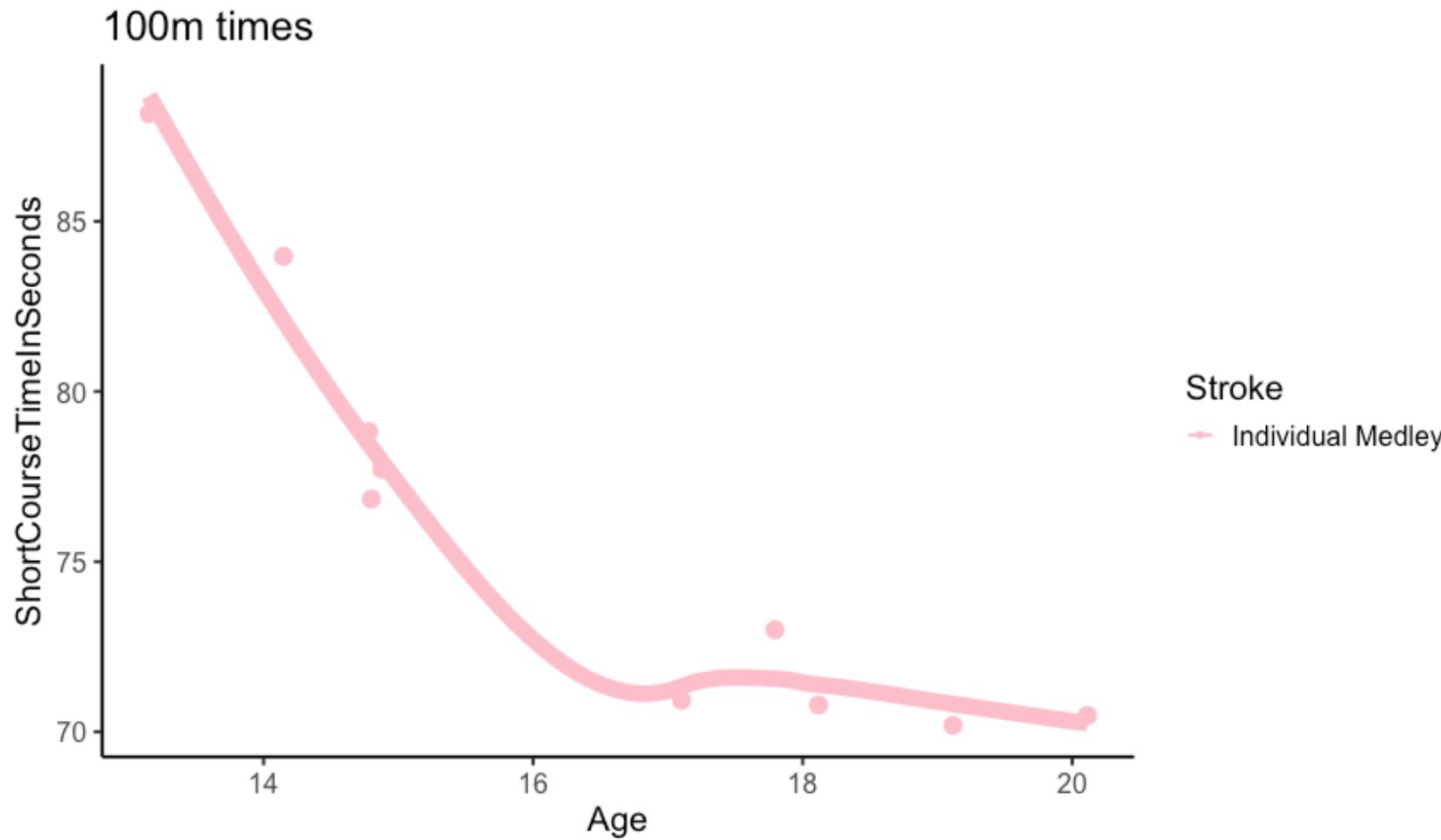
k-nearest neighbour

- Suppose our X training points are uniformly distributed in a p -dimensional unit hypercube.
- Let the nearest neighbour neighbourhood be a hypercube.
- What is the edge length E of the cube we need in order to get 0.01 of the data points?
 - to capture 0.01 of the points, we need the neighbourhood hypercube to have volume $V = 0.01$.
 - the volume of a hypercube is $V = E^p$, so $E = V^{1/p}$
 - for $p = 2, E = 0.1$
 - for $p = 5, E \approx 0.4$
 - for $p = 10, E \approx 0.63$.
- Even for relatively small p , one cannot think of this neighbourhood as local.

k -nearest neighbour

- An alternative view
 - Control the bias by fixing the edge length of the neighbourhood hypercube, to be $E = 0.1$ for example, so that we are always getting an estimate that is reasonably local.
 - Control the variance by ensuring the neighbourhood hypercube contains 10 points on average.
 - How large does n need to be such that these conditions are satisfied?
- The expected number of points in a hypercube with $E = 0.1$ is $n \times 0.1^p$
 - we need to know what n is to find $n \times 0.1^p = 10$
 - we have $n = 10^{p+1}$.
- Curse of dimensionality:
 - n needs to grow exponentially with p .

Other non-parametric methods



What about other non-parametric approaches?

- Most other non-parametric methods also require local training data points in order to make a good prediction
 - few assumptions are being made about the model, so we rely heavily on the training data to make good predictions.
- An unexpected (?) feature of a large p is that most of the points are found in the edges of the space.

What about other non-parametric approaches?

- Consider n uniformly distributed training points in a p -dimensional unit ball centred at the origin.
- The median distance from the origin to the closest data point is

$$\left(1 - \frac{1}{2}^{1/n}\right)^{1/p}$$

- for fixed n , as p gets bigger, the median distance increases towards 1.
- Points end up being closer to the boundary of the space than to any other data point
 - disaster for non-parametric methods, which rely on interpolating between training points.

Comparison to linear methods

- In general the number of points needed to control the error of non-parametric methods is exponential in p .
- Compare to linear regression
 - the expected excess risk is approximately $\sigma^2 \frac{p+1}{n}$
 - so n only needs to increase linearly in p to control the error
 - linear methods beat the curse of dimensionality by using restrictive assumptions.
- How can we use a flexible method as p grows?
- Idea:
 - we need to impose some kind of restriction that avoids the curse of dimensionality, but is still flexible in some useful way.

Naive Bayes

- *Naive Bayes* is a generative classification method.
- Recall: we wish to model the joint distribution $P(X, Y)$. There are two approaches we could use:
 - **Generative**. Choose a model for the factorisation

$$P(X, Y | \theta) = P_{X|Y}(X | Y, \theta_{X|Y})P_Y(Y | \theta_Y).$$

- **Discriminative**. We choose a model for the factorisation

$$P(X, Y | \theta) = P_{Y|X}(Y | X, \theta_{Y|X})P_X(X | \theta_X).$$

- $P_Y(Y | \theta_Y)$ can be thought of as a prior distribution on the model.
- $P_{X|Y}(X | Y, \theta_{X|Y})$ can be thought of as a model for the data within each class.

Naive Bayes

- Naive Bayes makes the assumption that the variables $X^{(j)}$ within each class are independent

$$P_{X|Y}(X | Y) = \prod_{j=1}^p P_{X^{(j)}|Y}(X^{(j)} | Y),$$

i.e. the $X^{(j)}$ are conditionally independent given Y .

- This is a drastic assumption, which may not be an accurate model for the data.
- However...
 - it mitigates the curse of dimensionality, since the conditional joint distribution of the p predictors depends only on the marginal distribution of each predictor
 - each marginal distribution may be estimated independently for each class.

Naive Bayes

- The assumption is such a simplification of reality that sometimes the approach is called "idiot's Bayes".
- It can often perform very well:
 - Hand, D. Idiot's Bayes: Not So Stupid after All? (2001). International Statistical Review, 69, 385-398.
- One way of extending naive Bayes is not to assume that the $X^{(j)}$ are independent, but to assume they factorise according to some set of conditional independence relationships given by a graphical model
 - a *Bayesian network classifier*
 - in practice this generalisation is not always worth it.

Example: Gaussian case

- In LDA $P_{X|Y}(X | Y)$ is chosen to be multivariate Gaussian.
- If we use the naive Bayes assumption with each $P_{X^{(j)}|Y}(X^{(j)} | Y)$ being univariate Gaussian, then $P_{X|Y}(X | Y)$ is a multivariate Gaussian with diagonal covariance matrix.
- In this case we can see naive Bayes as a restricted form of LDA, using $O(p)$ parameters, rather than $O(p^2)$
 - there is also an associated reduction in the computational complexity of training and test.
- The decision boundary is linear
 - but it doesn't need to be in general
 - often naive Bayes can have a "good" decision boundary even if the within class distributions are inaccurate.

Generative-discriminative pairs

- Recall that we could rewrite LDA as a discriminative classifier
 - we obtained a linear decision boundary
 - although LDA is fit differently to logistic regression.
- We may do the same with naive Bayes
 - this will point us to a general class of models that are similarly unaffected by the curse of dimensionality.
- Recall that to find the log odds of any class y_k compared to some (arbitrarily chosen) "base" class y_0 , we can use

$$\log \left[\frac{P_{Y|X}(Y = y_k \mid X = x)}{P_{Y|X}(Y = y_0 \mid X = x)} \right].$$

Naive Bayes → GAM

$$\begin{aligned}\log \left[\frac{P_{Y|X}(Y = y_k \mid X = x)}{P_{Y|X}(Y = y_0 \mid X = x)} \right] &= \log \left[\frac{P_{X|Y}(X \mid Y = y_k) P_Y(Y = y_k)}{P_{X|Y}(X \mid Y = y_0) P_Y(Y = y_0)} \right] \\ &= \log \left[\frac{P_Y(Y = y_k)}{P_Y(Y = y_0)} \right] + \log \left[\frac{\prod_{j=1}^p P_{X^{(j)}|Y}(X^{(j)} \mid Y = y_k)}{\prod_{j=1}^p P_{X^{(j)}|Y}(X^{(j)} \mid Y = y_0)} \right] \\ &= \log \left[\frac{P_Y(Y = y_k)}{P_Y(Y = y_0)} \right] + \sum_{j=1}^p \log \left[\frac{P_{X^{(j)}|Y}(X^{(j)} \mid Y = y_k)}{P_{X^{(j)}|Y}(X^{(j)} \mid Y = y_0)} \right] \\ &= \alpha^k + \sum_{j=1}^p f^{k,j}(X^{(j)}),\end{aligned}$$

- This has the structure of a *generalised additive model* (GAM)
 - although again, it is fit in a different way to a GAM.

Generalised additive models

- Recall a generalised linear model (GLM):

- model $\mathbb{E}_Y[Y | X] = g^{-1}(X^T \beta)$, or alternatively

$$g(\mathbb{E}_Y[Y | X]) = X^T \beta$$

- $X^T \beta$ known as the *linear predictor*
 - g known as the *link function*.

- A GAM uses

$$g(\mathbb{E}_Y[Y | X]) = \alpha + \sum_{j=1}^p f^j(X^{(j)})$$

- can be used for regression or classification
 - including multi-class classification
 - the functions f^j are chosen to be non-parametric functions such as splines.

Summary of large p

- For $p + 1 > n$
 - linear regression with least squares is not possible
 - shrinkage and/or variable selection is essential
 - most other approaches are not appropriate.
- For $p + 1$ large, but $< n$
 - linear regression can perform well with regularisation
 - non-parametric approaches cannot be used due to the curse of dimensionality
 - naive Bayes and GAMs can be feasible.

Last section of the module

- MCMC in high (and infinite!) dimensions
 - Langevin Monte Carlo
 - Hamiltonian Monte Carlo
 - preconditioned Crank–Nicolson algorithm.

Further reading

- ISLR chapter 3.
- ESL chapter 2, 9.
- PRML chapter 7.

MCMC for big data

ST420 Lecture 27

Richard Everitt

Recap

- We introduced big data problems early in the module
 - need to consider the effect of n and/or p being large.
- We introduced Bayesian statistics, and Monte Carlo, which is used for inference for Bayesian models
 - we have focussed on Bayesian linear regression and regularisation.
- We started to look at what we called "algorithmic" issues when we first looked at MCMC
 - so far, the challenge posed by p being large when designing MCMC for variable selection
 - the discrete space with 2^p states posed a difficult challenge, so we considered regularisation methods that have only continuous parameters (e.g. horseshoe priors).

Plan for the final 4 lectures

- This lecture:
 - some more depth in MCMC and Metropolis-Hastings
 - case of large p with random walk Metropolis-Hastings.
- Lecture 28:
 - Metropolis-adjusted Langevin algorithm (MALA)
 - large p and large n .
- Lecture 29:
 - Hamiltonian Monte Carlo
 - large p .
- Lecture 30:
 - preconditioned Crank-Nicolson.

MCMC for big data

- Statistical aspects of big data
 - large n makes things **easier** - means that we can fit more complex models
 - large p makes things **harder** - need more data to be able to use such models
- Both large n and large p have a "computational" impact
 - referring to the CPU cost, in terms of n and p of the number of operations we need to perform to run our algorithms.
- However, they also have an "algorithmic" impact
 - referring to the ability of our fitting algorithms - MCMC in this case - to accurately fit the model.

MCMC for big data

- What do we mean by the "algorithmic" issue, as distinct from the computational issue?
- We mean the impact of n and/or p on how long the fitting method takes to run
 - beyond simply the raw cost of working with (say) a big matrix representing the data.
- The key point is that most fitting algorithms are iterative
 - how do large n and/or p impact on how many iterations we need to run the method in order to obtain a good fit?
- In the end this will contribute to the computational cost of the approach.

Impact of large n on the number of iterations

- The posterior distribution will be more concentrated, since as we get more data, we have more information about the unknown parameters
 - this only affects the scale of the distribution, so it doesn't obviously affect the MCMC efficiency...
 - since we would need a smaller proposal variance to make the algorithm efficient, but the scale is unknown, so in this sense there is nothing "harder" about this situation.
- Other Monte Carlo methods might be more affected by the concentration
 - e.g. importance sampling where the prior is used as a proposal (not in this module).

Impact of large n on the number of iterations

- Also, the Bernstein-von Mises theorem says that as n increases, under some conditions, the posterior converges to a multivariate normal distribution (the sampling distribution of the maximum likelihood estimator)
 - do we really need to do Monte Carlo in this case?
 - can we simply estimate a Gaussian approximation to the posterior?
- We will not examine this issue further
 - although we will, in the next lecture, look at a way of trying to avoid the computational issue of needing to evaluating a likelihood with cost $O(n)$.

Impact of large p on the number of iterations

- The curse of dimensionality makes numerical integration infeasible as p grows.
- For MCMC, the important issue is how p growing affects the shape of the posterior distribution
 - this affects how efficiently we may explore the posterior distribution.
- We begin by giving an indication of what we are looking for in a "good" MCMC algorithm.
- Then we look at the shape of distributions in high dimensions.
- Followed by giving some results about the scaling of random walk Metropolis-Hastings with p .

Metropolis-Hastings

- The following algorithm has limiting distribution $\pi(\theta \mid y)$, starting at some initial point θ_0 .
-

Algorithm 1: Metropolis-Hastings

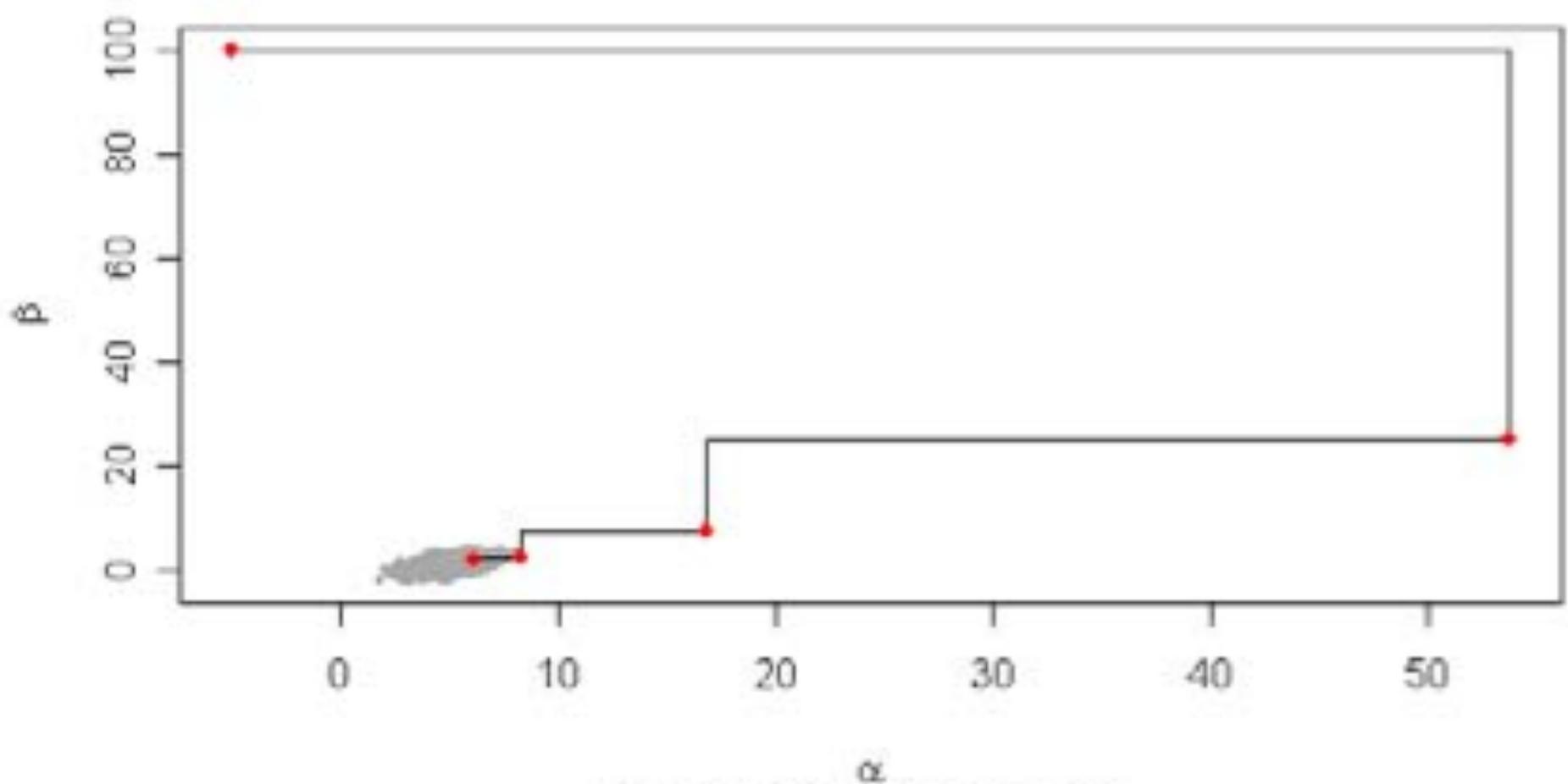
For $t = 1 : N$

- Simulate $\theta^* \sim q(\cdot \mid \theta_{t-1})$
 - Simulate $u \sim \mathcal{U}[0, 1]$
 - if $u < \min\left\{1, \frac{l(y \mid \theta^*) p(\theta^*) q(\theta_{t-1} \mid \theta^*)}{l(y \mid \theta_{t-1}) p(\theta_{t-1}) q(\theta^* \mid \theta_{t-1})}\right\}$
 - $\theta_t = \theta^*$
 - else
 - $\theta_t = \theta_{t-1}$
-

Checking the MCMC output

- Roughly speaking, what are we looking for when using MCMC in practice?
- **Convergence.** Is there any indication that we have not converged to the posterior?
 - examine trace plots
 - discard burn in
 - lack of convergence gives us a **bias**.
- **Efficiency.** Have we run the MCMC for long enough?
 - estimate the autocorrelation time (which allows us to estimate the Monte Carlo error) and plot the autocorrelation function
 - an inefficient chain gives use high **variance** estimates for a given computational budget.

Convergence: remainder of the problem



Convergence: ergodic theorems

- There are theoretical descriptions about the rate at which MCMC chains converge.
- For example, *geometric ergodicity*:
 - a Markov chain with stationary distribution π is geometrically ergodic if

$$\|P^N(\cdot | \theta_0) - \pi(\cdot)\|_{\text{TV}} \leq M(\theta_0)\rho^N,$$

for all $N \geq 1$ for some $\rho < 1$, where $M(\theta_0) < \infty$ for π almost everywhere θ_0 .

- Some chains can be shown to satisfy this condition
 - but this is not something that practitioners use
 - we usually cannot prove that this holds.

Convergence: in practice

- MCMC practitioners use “convergence diagnostics” that examine the points that have been generated, looking for signs of convergence.
- How can we identify when the chain has converged to equilibrium?
 - Does the chain seem stationary?
 - Do different starting values give similar results?
- There are no guarantees on these procedures - we can never be sure that we have converged simply by looking at this information, but we can diagnose when something has not converged.
- Suggests simulating multiple chains with different starting values.
- Remember it is convergence to a distribution, not to a point.

Convergence: diagnostics

- A popular approach originates in Gelman and Rubin (1992).
- Each MCMC chain should be stationary.
- Therefore the posterior densities from each chain - after burn in - should look the same.
- The variation between values within one chain should similar to the variation between values in different chains.
- More formally, we would expect the variation between values in one chain (W , within chain sum of squares) to be similar to the variation between values in different chains (B , between chain sum of squares).

Convergence: diagnostics

- “BGR” is the the Gelman and Rubin convergence statistic as modified by Brooks and Gelman (1998).
- For a single parameter it monitors:

$$R = \frac{\text{width of central 80\% credible interval based on all chains}}{\text{mean width of interval using each chain separately}}$$

calculated repeatedly for increasingly large fractions of the total iteration range.

- One should be concerned both with convergence of R to 1, and with convergence of both the pooled and within interval widths to stability.
- R is called the estimated scale reduction, or shrink factor.

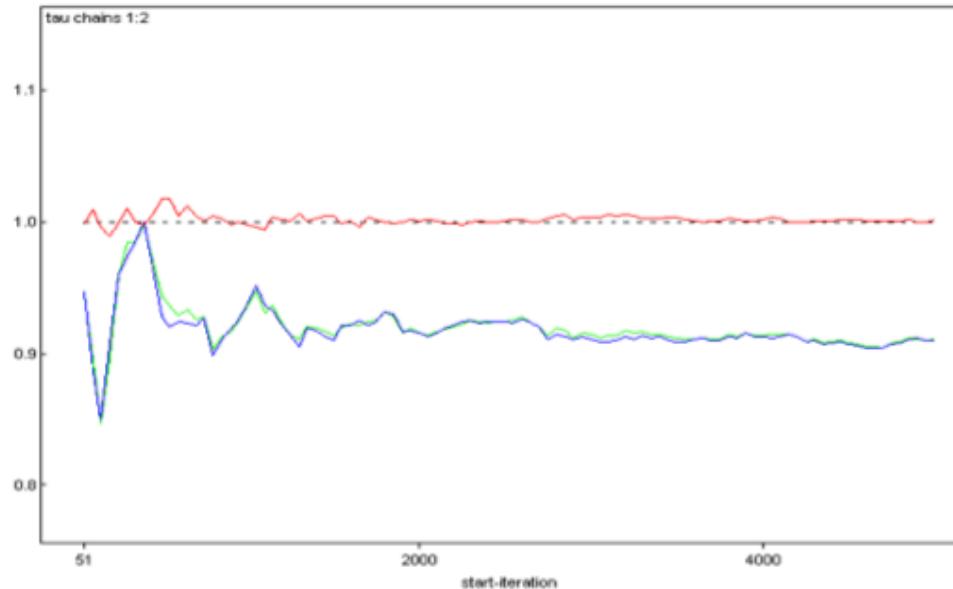
Convergence: diagnostics

Green: width of the 80% interval of pooled runs

Blue: average width of the 80% interval within runs

Red: R= ratio of pooled/within

Pooled and within are normalised to have an overall maximum of 1.



CHECK

Convergence of R to 1 (in practice <1.05)

Convergence of pooled and within run values to stability

Calculated for second half of iterations 1-100, 1-200, 1-300,...,1-2T so that start iterations are 51,101,151,...,T+1

Efficiency

- Once the chain has converged further iterations are required to obtain accurate posterior estimates.
- More iterations → more accurate posterior estimates.
- Monte Carlo error measures the statistical efficiency of the sample mean as an estimate of the theoretical posterior expectation.
- In this case the theory directly gives some ideas that are used in practice.

Efficiency: central limit theorem

- How does the Monte Carlo error in an MCMC compare to the case when points can be simulated independently?
- For independent points from a distribution of variance $\sigma^2 < \infty$ the CLT tells us that

$$\lim_{N \rightarrow \infty} \left(\widehat{\mathbb{E}_\pi[\theta]} - \mathbb{E}_\pi[\theta] \right) \rightarrow \mathcal{N}(0, \sigma^2/N).$$

- For MCMC in some cases we also have a CLT
 - if the chain is geometrically ergodic, and for some $\delta > 0$, $\mathbb{E}_\pi [\theta^{2+\delta}] < \infty$, then

$$\lim_{N \rightarrow \infty} \left(\widehat{\mathbb{E}_\pi[\theta]} - \mathbb{E}_\pi[\theta] \right) \rightarrow \mathcal{N}(0, \tau\sigma^2/N),$$

where

$$\tau = \sum_{k=-\infty}^{\infty} \rho_k$$

is the *integrated autocorrelation time* (IAT), with ρ_k the autocorrelation function at lag k ^{18/34}

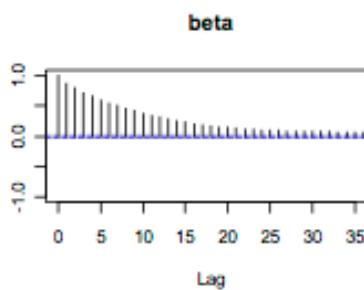
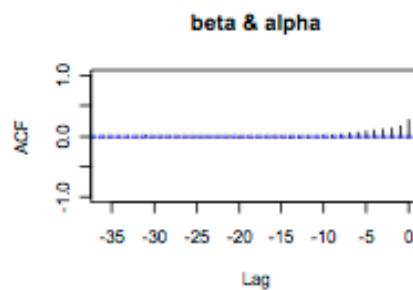
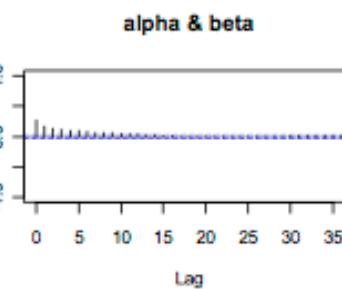
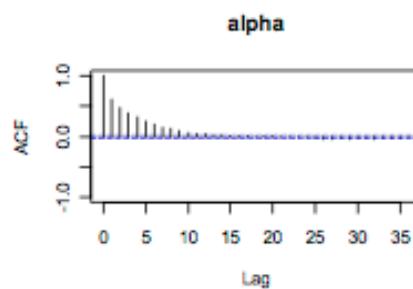
Efficiency: autocorrelation

- τ provides a measure of the efficiency of the relative statistical efficiency of an MCMC algorithm, compared to independent sampling from the posterior
 - note that for independent points, we have $\rho_0 = 1$ and $\rho_k = 0$ for $k \neq 0$, thus $\tau = 1$
 - when points from our MCMC chain are less dependent, we will obtain lower variance estimators.
- τ can be estimated from MCMC output
 - it is notoriously difficult to estimate accurately
 - but the order of magnitude of the estimate is usually informative.
- Often practitioners plot the estimated autocorrelation function ρ_k as a function of k .

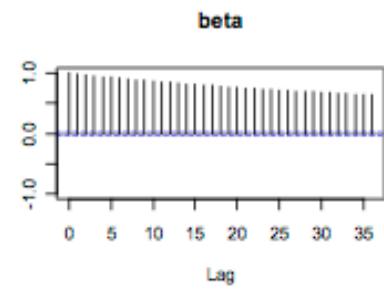
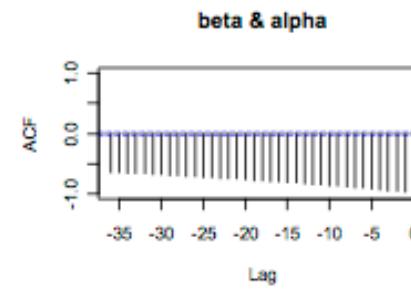
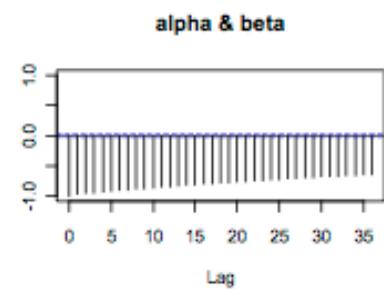
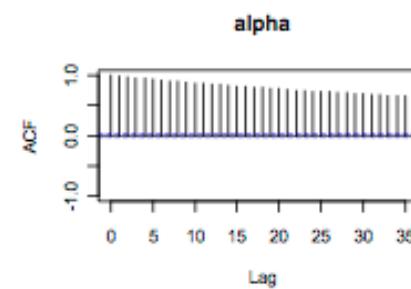
Efficiency: autocorrelation function

- This autocorrelation for each variable can be examined from MCMC output (here for a model with parameters α and β).

Not highly correlated



Highly correlated



Distributions in high dimensions

- We wish to estimate expectations with respect to π , e.g.

$$\mathbb{E}_\pi[\theta] = \int_\theta \theta \pi(\theta) d\theta,$$

where π is the density and we can think of $d\theta$ as being the volume.

- In Monte Carlo we draw points to represent the mass $\pi(\theta)d\theta$ of the probability distribution.
- The behaviour of volume in high dimensions leads to some results that are possibly surprising
 - we have already seen some such properties when discussing the curse of dimensionality.
- For this lecture we need
 - to know the amount of volume outside of a sphere
 - to understand what high-dimensional Gaussians look like.

Volume outside of a sphere

- The volume of a sphere with radius r in p dimensions is, where Γ is the gamma function,

$$\frac{\pi^{\frac{p}{2}}}{\Gamma\left(\frac{p}{2} + 1\right)} r^p.$$

- Recall that where k is a positive integer, $\Gamma(k) = (k - 1)!$
 - thus we see that the the volume goes to zero.

Volume outside of a sphere

- However the ratio of the volume of a sphere with radius δ more than r to the volume of a sphere with radius r is

$$\frac{\frac{\pi^{\frac{p}{2}}}{\Gamma\left(\frac{p}{2}+1\right)} (r+\delta)^p}{\frac{\pi^{\frac{p}{2}}}{\Gamma\left(\frac{p}{2}+1\right)} r^p} = \left(\frac{r+\delta}{r}\right)^p.$$

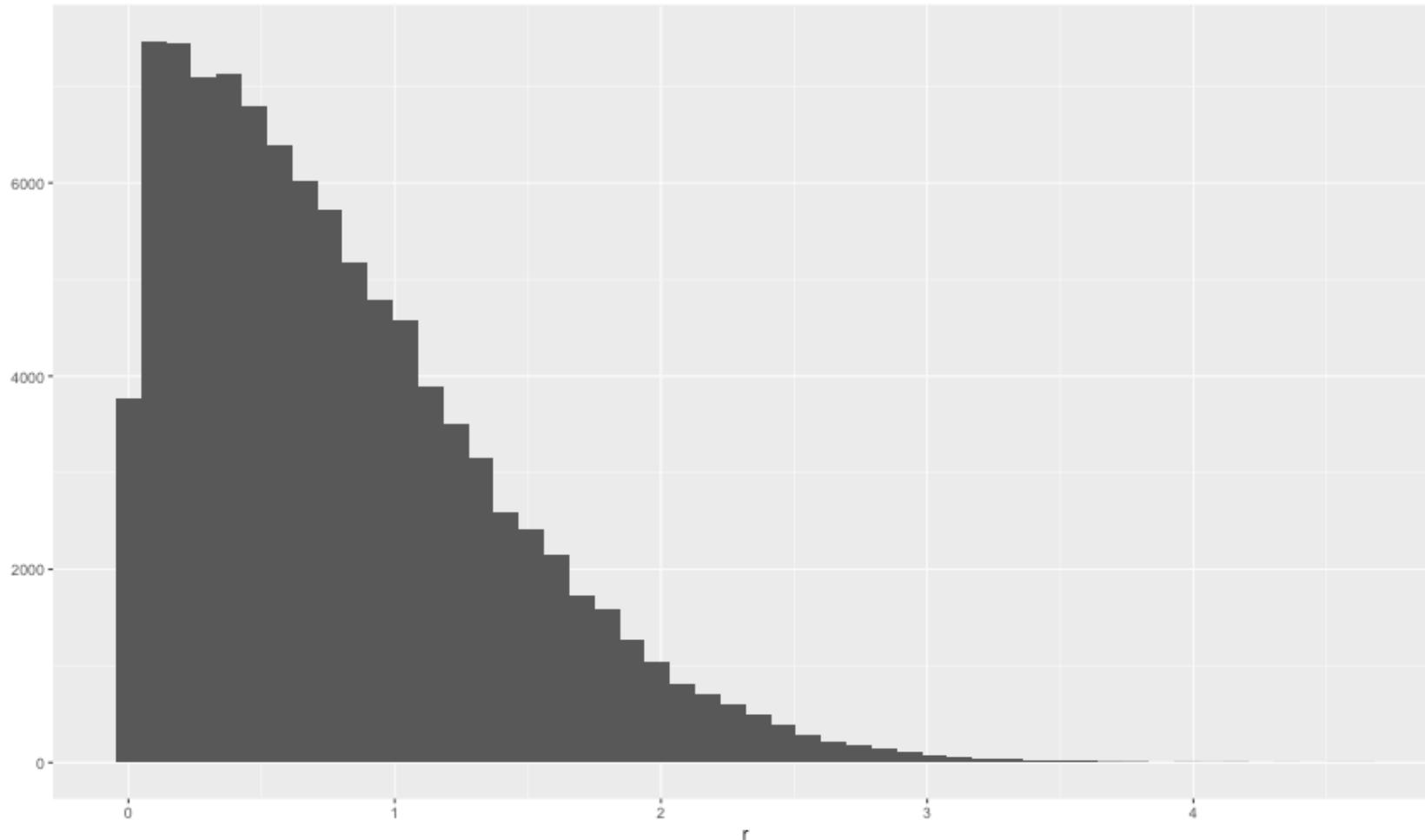
- We see that there is exponentially more volume just outside of the sphere than within the sphere.

High-dimensional Gaussians

- Let's study empirically what the mass of Gaussians looks like in different dimensions.

```
library(ggplot2)
x = rnorm(100000, 0, 1)
r = abs(x)
```

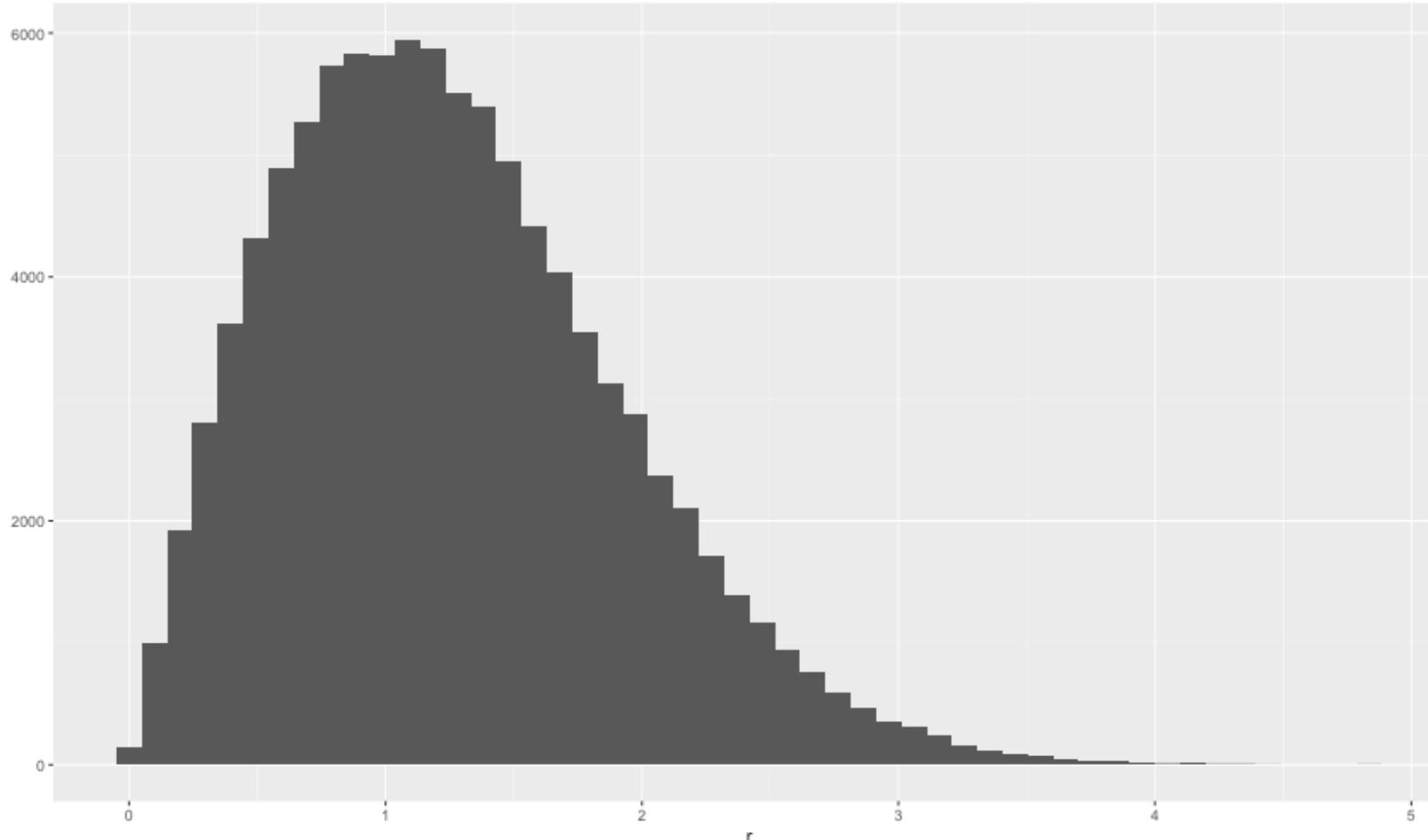
High-dimensional Gaussians



High-dimensional Gaussians

```
library(mvtnorm)
p = 2
x = rmvnorm(100000, rep(0, p), diag(p))
r = sqrt(rowSums(x^2))
```

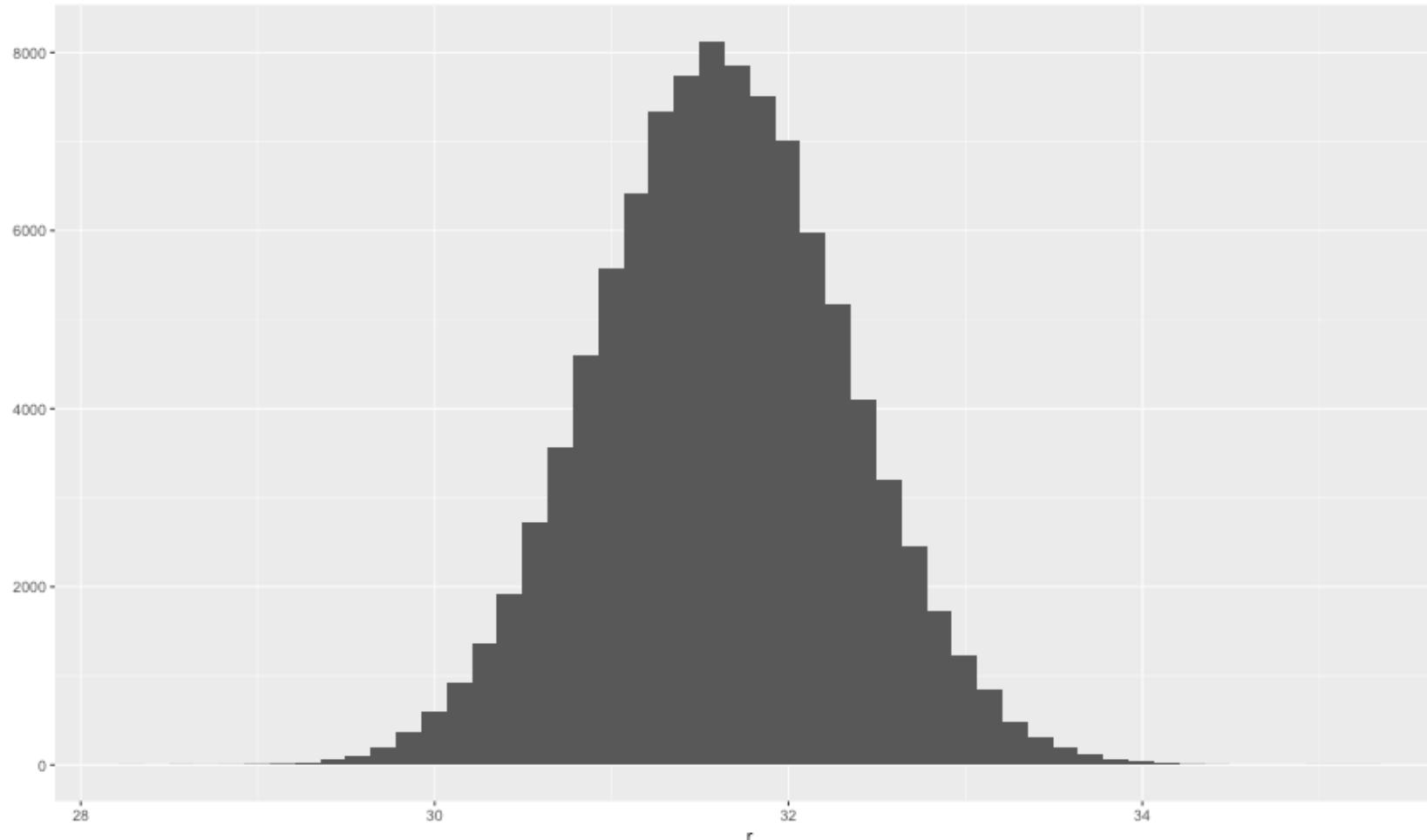
High-dimensional Gaussians



High-dimensional Gaussians

```
library(mvtnorm)
p = 1000
x = rmvnorm(100000, rep(0, p), diag(p))
r = sqrt(rowSums(x^2))
```

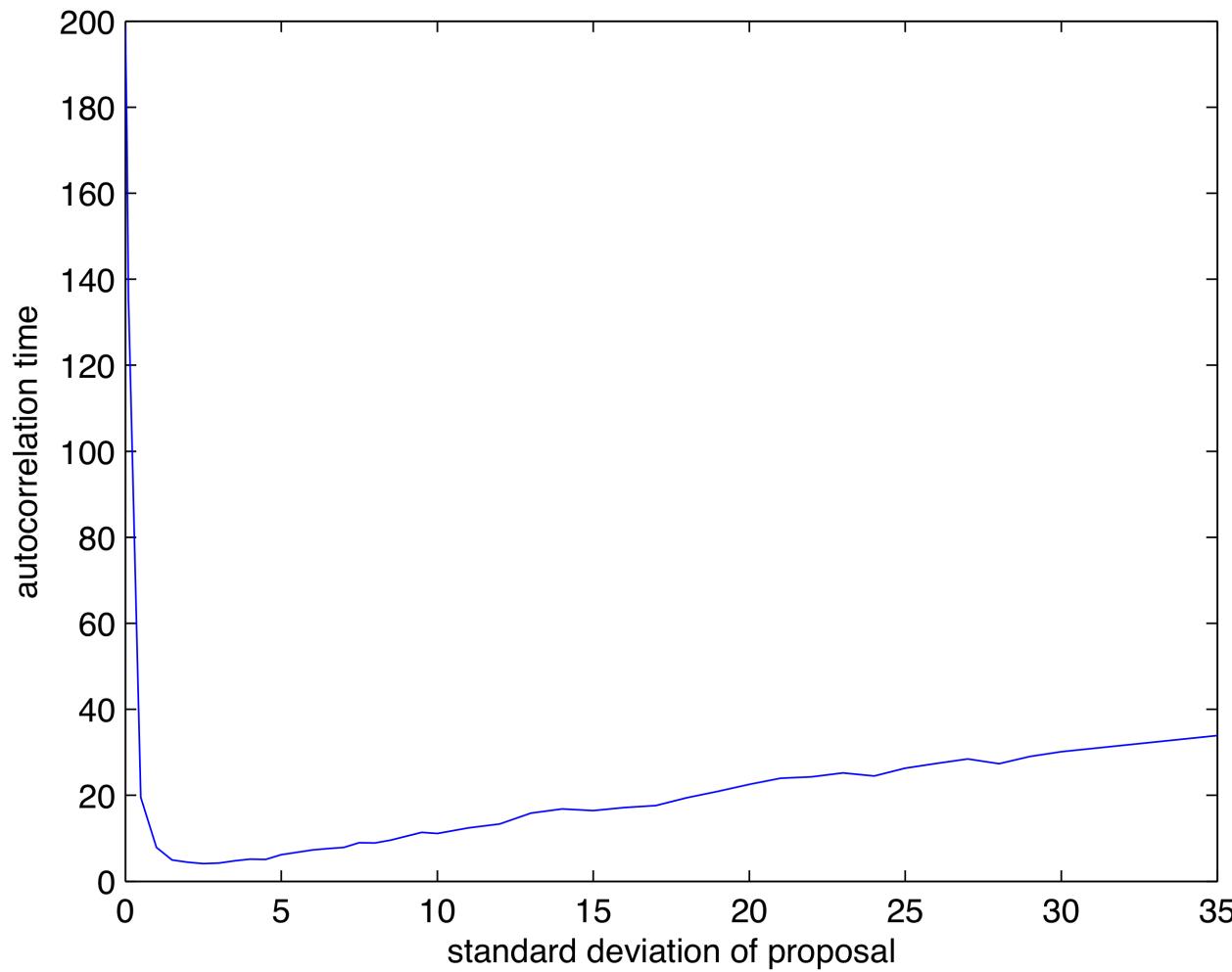
High-dimensional Gaussians



High-dimensional Gaussians

- We have that
 - the density of the Gaussian is concentrated around the mode at the origin
 - but there is more volume further away from the origin.
- The result is that the mass is found in a thin shell some distance from the origin.
- When we do MCMC, this is the region we need points from.
- Note in particular that the mode is a long way from this mass
 - is the mode (the MAP estimate in Bayes) an appropriate statistic for representing the posterior
 - recall that this is what ridge and lasso are using.

Efficiency of random walk Metropolis-Hastings



Efficiency of random walk Metropolis-Hastings

- The previous slide tells us that when the target and proposal are both univariate Gaussian (here the standard deviation of the target is 1), we should not choose the proposal standard deviation to be too small, or too large
 - it can be shown that the optimal acceptance rate in this case is 0.44.
- It doesn't tell us what the impact of the dimension p of θ is.
- Consider how what we know about target distributions in high dimensions affects the optimal proposal variance.

Efficiency of random walk Metropolis-Hastings

- Consider a Gaussian target distribution
 - in high dimensions the mass is in a thin shell around the mode.
- As the dimension increases, the volume on the exterior of the shell is very large, and most of our proposals will be here (i.e. in the tails of the distribution).
- However, the density of these points is very small, so we have many rejections.
- To counter this effect, we can make the proposal variance smaller to obtain fewer rejections.

Random walk Metropolis-Hastings with large p

- Roberts, Gelman and Gilks (1997) find, under certain conditions, that
 - the optimal proposal variance is $O(p^{-1})$
 - the optimal τ is proportional to p
 - $O(p)$ steps are needed to reach a nearly independent point.
- Recall that the computational cost of a single iteration of Metropolis-Hastings is $O(p)$
 - therefore the method needs $O(p^2)$ computation time.
- Can we do better?
 - next time.

The Metropolis-adjusted Langevin algorithm

ST420 Lecture 28

Richard Everitt

Recap

- We are examining the effect of the dimension p of the parameter space on the number of iterations for which we must run an MCMC algorithm
 - we have referred to this as the "algorithmic" cost
 - it is a particular contribution to the total computational cost.
- We have seen that the number of iterations we must run random walk Metropolis-Hastings to obtain one effectively independent point is $O(p)$
 - we obtain this since we have that the optimal autocorrelation time τ is of the order $O(p)$, and that the effective number of independent points for an MCMC is N/τ .
- We have seen why using a random walk may not be appropriate in high dimensions.

Distributions in high dimensions

- In high dimensions, the mass of a distribution will tend to be found in a thin shell around the mode.
- How can we set up an MCMC to efficiently explore this thin shell?
 - random walk is not the answer.
- Idea:
 - use the gradient of the target distribution to help us
 - very similar to the idea of "stochastic approximation", also known as "stochastic optimisation".

Stochastic approximation

- First described in Robbins and Monroe (1951) as a method for finding the maximum of functions of which we can only obtain noisy observations.
- To keep common notation with the Langevin algorithm described later, we consider the use of stochastic approximation for finding the MAP estimate
 - use $p(\theta)l(y | \theta)$ as the function to be maximised.
- At each step, use

$$\theta_{t+1} = \theta_t + \frac{\epsilon_t}{2} (\nabla \log p(\theta_t) + \nabla \log l(y | \theta_t)),$$

where ϵ_t is some decreasing sequence of "gains".

- The idea is to use the gradient to help climb uphill and find the mode.

Langevin dynamics

- Our aim is to use a similar idea within MCMC
 - in low dimensions, regions of high density are a good guide for helping us find the probability mass
 - in higher dimensions, we are making it less likely to propose in the tails.
- Our starting point is the stochastic differential equation (SDE)

$$d\theta_t = \frac{1}{2} (\nabla \log p(\theta_t) + \nabla \log l(y | \theta_t)) dt + dW_t,$$

where W_t is the Wiener process.

- This SDE defines Langevin dynamics.

Langevin dynamics

- What does this SDE mean?
- Informally...

$$\theta_{t+\epsilon} - \theta_t = \int_{u=t}^{t+\epsilon} \frac{1}{2} (\nabla \log p(\theta_t) + \nabla \log l(y | \theta_t)) du + \int_{u=t}^{t+\epsilon} dW_u$$

- roughly speaking, this says that within a small time interval ϵ , θ_t changes by an increment that is normally distributed with expectation

$$\frac{\epsilon}{2} (\nabla \log p(\theta_t) + \nabla \log l(y | \theta_t))$$

and variance ϵ (using the properties of the Wiener process).

Langevin dynamics

- Langevin dynamics originates in the physics literature.
- Why is it useful to us?
 - since the process has an invariant distribution of $\pi(\theta \mid y)$
 - note that we avoided needing the normalising constant of π since it is irrelevant to the gradient.
- If we could exactly simulate Langevin dynamics, we would obtain points from the posterior π
 - the issue is that we cannot perform this (continuous time) simulation.
- Use an SDE solver to simulate the dynamics approximately.

Euler-Maruyama simulation

- We gave a heuristic interpretation of the increments of the SDE in terms of a normal distribution.
- Idea of Euler-Maruyama:
 - using some $\epsilon > 0$
 - set $\theta_{t+1} \sim q(\theta_{t+1} | \theta_t)$ where
- This is only an approximate simulation of the dynamics
 - in general it will not have π as its limiting distribution.

Metropolis-adjusted Langevin algorithm

- However, we could use this choice of $q(\theta_{t+1} \mid \theta_t)$ as a proposal in Metropolis-Hastings
 - proposals are accepted with the usual probability.
- This method is called the *Metropolis-adjusted Langevin algorithm* (MALA)
 - has the benefits of Langevin dynamics
 - can be implemented in practice
 - has the correct invariant distribution.
- Dependence on dimension
 - the optimal proposal variance is $O(p^{-1/3})$
 - the optimal τ is proportional to $p^{1/3}$
 - $O(p^{1/3})$ steps are needed to reach a nearly independent point
 - overall cost is $O(p^{4/3})$.

Preconditioning

- As in optimisation algorithms, *preconditioning* can be used to speed up convergence
 - the idea is to make the space look more "circular" - if π varies more in some dimensions than others.
- Use the proposal

$$q(\theta_{t+1} \mid \theta_t) = \mathcal{N}\left(\theta_t + \frac{\epsilon A}{2} (\nabla \log p(\theta_t) + \nabla \log l(y \mid \theta_t)), \epsilon\right),$$

where A is some positive-definite preconditioning matrix of dimension $p \times p$.

- Finding a good preconditioning matrix can be expensive, and thus may result in an overall increase in computational cost.

Big data case

- Note the strong link between stochastic approximation using the Robbins-Monro scheme

$$\theta_{t+1} = \theta_t + \frac{\epsilon_t}{2} (\nabla \log p(\theta_t) + \nabla \log l(y | \theta_t))$$

and the *unadjusted Langevin* update

$$\theta_{t+1} = \theta_t + \frac{\epsilon}{2} (\nabla \log p(\theta_t) + \nabla \log l(y | \theta_t)) + \varepsilon_t,$$

where $\varepsilon_t \sim \mathcal{N}(0, \epsilon)$.

Big data case

- In stochastic approximation it is common practice to use a "noisy" gradient estimate of $\nabla \log l_n(y | \theta)$, where l_n denotes the likelihood based on n data points
 - instead the likelihood l_η based on $\eta < n$ data points is used, in the following update

$$\theta_{t+1} = \theta_t + \frac{\epsilon_t}{2} \left(\nabla \log p(\theta_t) + \frac{n}{\eta} \nabla \log l_\eta(y | \theta_t) \right)$$

- the scaling n/η is used due to the log-likelihood $\log l_\eta$ being a sum of η terms.

Stochastic Gradient Langevin Dynamics

- *Stochastic Gradient Langevin Dynamics* uses the update

$$\theta_{t+1} = \theta_t + \frac{\epsilon_t}{2} \left(\nabla \log p(\theta_t) + \frac{n}{\eta} \nabla \log l_\eta(y | \theta_t) \right) + \varepsilon_t,$$

where $\varepsilon_t \sim \mathcal{N}(0, \epsilon_t)$.

- The idea is to
 - use a decreasing sequence of ϵ_t to begin with, to quickly converge using stochastic approximation
 - have ϵ_t converge to some value greater than zero, such that later in the algorithm, the method will approximately follow Langevin dynamics.
- We are moving outside of the standard MCMC framework, so do not have the same convergence guarantees
 - but the algorithm can work in practice in some cases.

Summary

- We have seen how Langevin dynamics can help MCMC.
- Next time we see an enhanced version
 - Hamiltonian MCMC.

Hamiltonian Monte Carlo

ST420 Lecture 29

Richard Everitt

Recap

- We are examining the effect of the dimension p of the parameter space on the number of iterations for which we must run an MCMC algorithm
 - we have referred to this as the "algorithmic" cost
 - it is a particular contribution to the total computational cost.
- We have seen that the number of iterations we must run the following algorithms to obtain one effectively independent point is
 - $O(p)$ for random walk Metropolis-Hastings, which gives an overall computational cost of $O(p^2)$;
 - $O(p^{1/3})$ for the Metropolis-adjusted Langevin algorithm, which gives an overall computational cost of $O(p^{4/3})$.

This lecture

- We have seen that the random walk does not scale well with increasing dimension.
- MALA has better properties, but we can improve on this.
- In this lecture we see how HMC is a construction that is designed to directly tackle the problem we posed in lecture 26
 - simulating from high-dimensional distributions
 - uses intelligent proposals described by "Hamiltonian dynamics"
 - proposed as "hybrid Monte Carlo" (a hybrid between Monte Carlo and Hamiltonian dynamics)
 - now usually known as "Hamiltonian Monte Carlo:
 - "[MCMC from Hamiltonian dynamics](#)" by Radford Neal is a very thorough review of the topic.

Constructing a proposal: dynamics of a ball

- For random walk, we found that we needed to decrease the proposal variance as the dimension increased.
- We would like to have proposals that move a long way, but still have a good probability of acceptance
 - we need a proposal that follows the mass of the distribution.
- Think of the negative log of the target distribution, and consider the idea of setting a ball rolling around this surface
 - someone with a background in physics could describe the dynamics of this ball.
- Idea:
 - give the ball a push in a random direction
 - follow the dynamics of the ball for a while
 - use this as the proposal.

Hamiltonian dynamics

- Hamiltonian mechanics is an abstract formulation of classical mechanics (i.e. equations of motion, etc).
- It describes a system involving two time-evolving vectors θ and v , each of dimension p .
- The *Hamiltonian* $H(\theta, v)$ describes the time evolution of the system, through Hamilton's equations

$$\frac{d\theta_i}{dt} = \frac{\partial H}{\partial v_i} \quad \frac{dv_i}{dt} = - \frac{\partial H}{\partial \theta_i}$$

for $i = 1, \dots, p$.

- Note that physicists would be very annoyed by the notation here, where the vectors are called q and p instead of θ and v .
- This is very abstract
 - what do these equations mean?

Hamiltonian dynamics: total energy

- In the use of this technique in MCMC, we use these dynamics to describe a frictionless ball rolling around the negative log of the posterior distribution, subject to a gravitational pull.
- The vector θ denotes the position of the ball, and the vector v its momentum
 - recall that momentum is equal to mass times velocity
 - for simplicity we will take the mass of the ball to be 1, which means that momentum equals velocity.
- $H(\theta, v)$ represents the total energy of the ball

$$\underbrace{H(\theta, v)}_{\text{total energy}} = \underbrace{U(\theta)}_{\text{potential energy}} + \underbrace{K(v)}_{\text{kinetic energy}} .$$

Hamiltonian dynamics: potential energy

- Recall from classical mechanics that gravitational potential energy U is equal to mgh , where m is the mass of the ball, g is the gravitational field, and h is the height.
- For simplicity, we simply set m and g to be equal to 1.
- Therefore we simply take $U(\theta)$ to be the height of the ball at θ

$$U(\theta) = -\log(\pi(\theta \mid y)).$$

- For example, $U(\theta) = \theta^2$ would correspond to a Gaussian with zero mean.

Hamiltonian dynamics: kinetic energy

- Recall from classical mechanics that kinetic energy K is equal to a half times mass times velocity squared.
- In our case (with $m = 1$, momentum equals velocity). We obtain, in the univariate case, $K = v^2/2$.
- We are looking at the multivariate case, which gives $K(v) = v^T v/2$.

Hamiltonian dynamics: Hamiltonian

- The Hamiltonian in our case is given by

$$H(\theta, v) = -\log(\pi(\theta | y)) + v^T v / 2.$$

- Hamilton's equations in our case are given by

$$\frac{d\theta}{dt} = v \quad \frac{dv}{dt} = \nabla \log(\pi(\theta | y)).$$

- These make sense!
 - the rate of change of position is given by the velocity
 - the rate of change of velocity is given by the gradient of the surface.
- To construct a proposal for use in MCMC, we will simply simulate forwards from these dynamics for some time t
 - this simulation defines a deterministic function R_t , mapping $(\theta, v) \mapsto (\theta^*, v^*)$.

Hamiltonian dynamics: properties

- What did we gain from the abstract formulation, rather than simply working out this formulation from classical mechanics?
- Hamiltonian dynamics has some nice mathematical properties, that are particularly useful when constructing MCMC updates (here we follow Neal (2011)).
- **Reversibility.** There is an inverse to R_t , and this can be defined in terms of R_t . We have that R_t^{-1} is given by
 - taking the negative of the velocity (to make the ball go backwards)
 - applying R_t (running the dynamics for time t)
 - taking the negative of the velocity of the result (to make the ball "face" back in the direction it was originally)
 - we need this property for the dynamics to have π as the invariant distribution.

Hamiltonian dynamics: properties

- **Conservation of the Hamiltonian.** The dynamics do not change the value of H - the total energy of the ball is conserved.
 - this property is crucial in ensuring that the acceptance probability is high
 - soon we will define the joint distribution of θ and v in terms of H - the conservation of H under the dynamics will mean that (θ, v) has the same density as (θ^*, v^*) .
- **Volume preservation.** Hamiltonian dynamics preserves volume in the space of (θ, v) . This means that no Jacobian is needed when calculating the acceptance probability of a move (as it is in some other methods).

Hamiltonian Monte Carlo

- We now have most of the ingredients needed to define Hamiltonian Monte Carlo.
- We proceed as follows
 - define a joint distribution on (θ, v) such that we can run Hamiltonian dynamics on it in order to obtain points from π
 - describe how to deal with the fact that we cannot simulate Hamiltonian dynamics exactly.

Hamiltonian Monte Carlo: joint distribution

- Define a joint distribution on (θ, v) as follows

$$\begin{aligned}\pi_{\theta,v}(\theta, v) &\propto \exp(-H(\theta, v)) \\ &= \exp(-U(\theta)) \exp(-K(v)) \\ &= \exp(-(-\log(\pi(\theta | y)))) \exp(-v^T v / 2) \\ &= \pi(\theta | y) \exp(-v^T v / 2).\end{aligned}$$

- We see that the joint distribution on (θ, v) has $\pi(\theta | y)$ as its marginal, and that we have a Gaussian distribution on v
 - we could choose a different covariance for this Gaussian distribution on v - this would correspond to using a different mass for the ball in the potential energy.

Using Hamiltonian dynamics as an MCMC move

- “A Note On Metropolis-Hastings Kernels For General State Spaces”, Tierney (1998) gives the Metropolis-Hastings acceptance probability for a volume preserving deterministic move T that is an involution, i.e. where, in our case,

$$T(T(\theta, v)) = (\theta, v).$$

The acceptance probability is given by

$$\min \left\{ 1, \frac{\pi(T(\theta, v))}{\pi(\theta, v)} \right\}.$$

- We define T to be the composition of applying Hamiltonian dynamics $R_t(\theta, v)$, then taking the negative of the velocity component.

Using Hamiltonian dynamics as an MCMC move

- Then, using the conservation of the Hamiltonian, the acceptance probability of applying Hamiltonian dynamics to the joint target is given by

$$\min \left\{ 1, \frac{\pi_{\theta, v}(T(\theta, v))}{\pi_{\theta, v}(\theta, v)} \right\} = \min \left\{ 1, \frac{\exp(-H(T(\theta, v)))}{\exp(-H(\theta, v))} \right\} = 1,$$

which means that we would always accept such a move!

- This means that we can potentially make very large moves, as long as we choose appropriately the time for which we simulate the dynamics
 - too short, and we will not move far
 - too long, and it is possible that we end up where we started!
- The move using the dynamics is alternated with simulating a new velocity exactly from the target distribution for v
 - so that we change the direction of the trajectories at different iterations.

Approximating Hamiltonian dynamics

- We cannot simulate Hamiltonian dynamics exactly
 - we must use some solver, just as we did for the Langevin method.
- We use the "leapfrog" method to approximately simulate the dynamics
 - this produces a discretised trajectory that approximates the continuous dynamics
 - the transformation produced using this approach is also reversible and volume preserving.

Approximating Hamiltonian dynamics

- However the Hamiltonian is not exactly conserved
 - this means that the acceptance probability is not 1.
- Let T be the transformation given by the leapfrog method, and $(\theta^*, v^*) = T(\theta, v)$, then the acceptance probability is

$$\min \left\{ 1, \frac{\pi(T(\theta, v))}{\pi(\theta, v)} \right\} = \min \left\{ 1, \exp(-H(\theta^*, v^*) + H(\theta, v)) \right\},$$

- note that, as in standard Metropolis-Hastings, we can use $p(\theta)l(y | \theta)$ in place of $\pi(\theta | y)$, since the normalising constant $p(y)$ cancels.
- When implementing the leapfrog method, we need $\nabla \log(\pi(\theta | y))$. This is given by $\nabla \log p(\theta_t) + \nabla \log l(y | \theta_t)$ as in the previous lecture.

HMC properties

- Dependence on dimension
 - the optimal τ is proportional to $p^{1/4}$
 - $O(p^{1/4})$ steps are needed to reach a nearly independent point
 - overall cost is $O(p^{5/4})$
 - this beats both random walk and MALA.
- The Langevin simulation turns out to be equivalent to doing one step of the leapfrog method
 - so we can see HMC as an extension to MALA.
- The tuning of HMC makes a big difference to the performance
 - much research is devoted to automating this tuning
 - the "no u-turn sampler" (NUTS) is a significant contribution.

HMC in action



Summary

- Limitations
 - continuous variables only (e.g. can be used with horseshoe priors, but not variable selection)
 - doesn't help with multi-modal targets.
- HMC gives us a way of making large moves with a high acceptance rate, even when p is large.
- For many problems this is the state of the art approach.
- Implemented in the software "Stan", which uses automatic differentiation in order to make it easy to use for large classes of models.

MCMC on function spaces

ST420 Lecture 30

Richard Everitt

Recap

- We have been examining the effect of the dimension p of the parameter space on the number of iterations for which we must run an MCMC algorithm
 - we have referred to this as the "algorithmic" cost
 - it is a particular contribution to the total computational cost.
- We have seen that the number of iterations we must run the following algorithms to obtain one effectively independent point is
 - $O(p)$ for random walk Metropolis-Hastings, which gives an overall computational cost of $O(p^2)$;
 - $O(p^{1/3})$ for the Metropolis-adjusted Langevin algorithm, which gives an overall computational cost of $O(p^{4/3})$
 - $O(p^{1/4})$ for the Hamiltonian Monte Carlo, which gives an overall computational cost of $O(p^{5/4})$.

Can we take p to infinity?

- Firstly, would might we want to?
- Consider the problems of Bayesian nonparametric regression, classification, or density estimation
 - regression: estimate a function of X that predicts a continuous response Y
 - classification: estimate a function of X that predicts a discrete response Y
 - density estimation: estimate the probability density of observations of the variable Y .

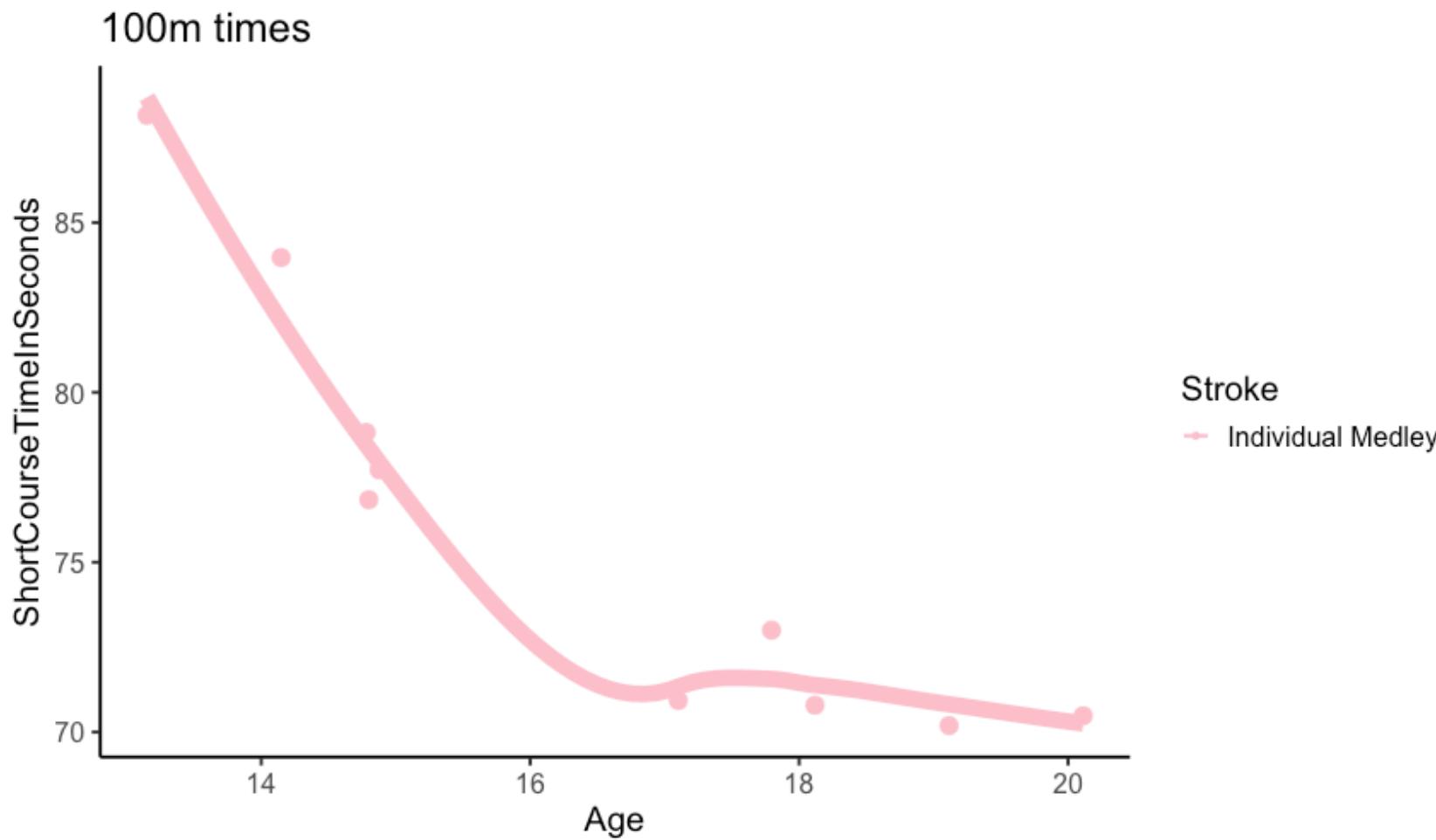
Can we take p to infinity?

- Each case involves estimating a random function
 - regression: the regression of Y on X
 - classification: the separating function that is used in, for example, a perceptron
 - density estimation (a type of unsupervised learning): a function f that we can use in the following formula to define a probability density ρ on some set $A \subseteq \mathcal{Y}$

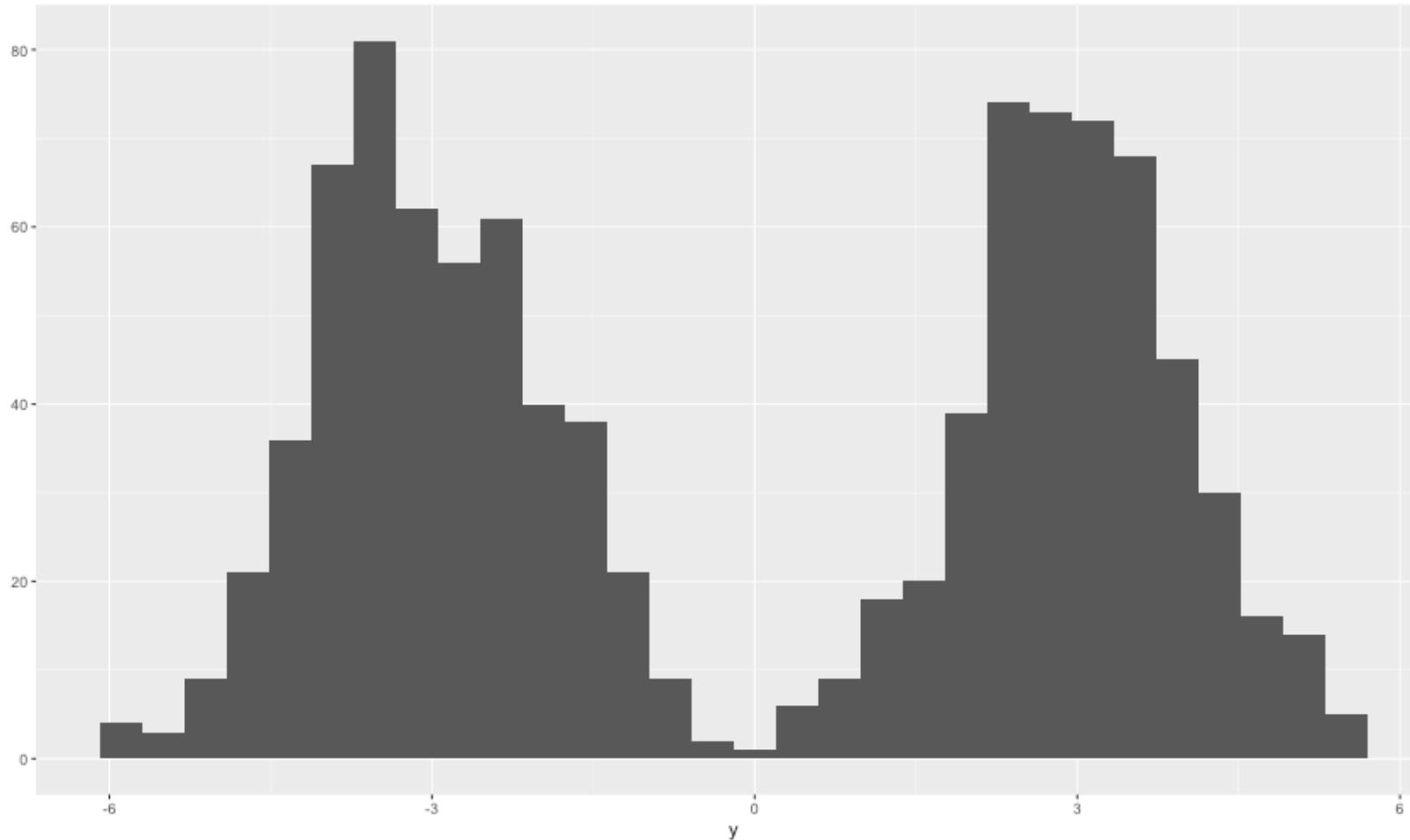
$$\rho(y) = \frac{\exp(f(y))}{\int_A \exp(f(s))ds}$$

over observed data $\{y_i\}_{i=1}^{\infty}$.

Example: non-parametric regression



Example: density estimation (data)



Random functions

- How might we run an MCMC to estimate a function f ?
 - restrict ourselves to the case to a function on \mathbb{R} for simplicity.
- In Bayesian statistics, we consider the things we don't know to be random
 - thus we treat f as a random function.

Random functions

- The Karhunen–Loève theorem gives a representation of a random function, on a bounded domain $[a, b]$, as an infinite linear combination of orthogonal functions

$$f(y) = \sum_{j=1}^{\infty} \beta^j \phi^j(y),$$

where $\{\beta^j\}_{j=1}^{\infty}$ are random variables

- $\beta = \{\beta^j\}_{j=1}^{\infty}$ is the representation of the function that we use (we used a similar idea when looking at kernel methods)
- over all possible basis expansions, the Karhunen–Loève theorem gives the smallest mean squared error when the expansion is truncated.

Bayesian inference of random functions

- Find

$$\pi(f \mid y) \propto l(y \mid f)p(f).$$

- We will use a Gaussian process prior $p(f)$ on functions
 - priors on functions, whose marginal distributions, when evaluated at any finite set of m points, are \mathbb{R}^m -valued Gaussians
 - in a Gaussian process the variables $\{\beta^j\}_{j=1}^\infty$ are independent and Gaussian distributed.

Bayesian inference of random functions

- The likelihood $l(y | f)$ is defined differently depending on the particular situation under consideration
 - for example, in regression, our model might be that the response is Gaussian with mean given by f , so the likelihood is given by the product of Gaussian distributions with mean $f(x_i)$ and some variance
 - for density estimation, we have

$$l(y | f) = \prod_{i=1}^n \rho(y_i).$$

MCMC on function spaces

- We wish to simulate from $\pi(f \mid y)$.
- Recall that our parameterisation of f is the infinite set of coefficients $\{\beta^j\}_{j=1}^\infty$
 - clearly we cannot infer a posterior over this infinite set in practice
 - let's consider some truncation of the set up to the first p terms $\{\beta^j\}_{j=1}^p$.
- What will happen if we use an MCMC to explore the posterior of these p variables?
- Whichever method we choose (random walk, MALA, HMC), for any finite step size, the acceptance rate will go to zero as p goes to infinity
 - it may be the case that some small value of p is "enough" accuracy for some applications, but for others we may need to know f to a high degree of accuracy, and in these cases it may make MCMC infeasible.

MCMC on function spaces

- Idea:
 - instead of looking at an MCMC on the finite dimensional parameterisation, construct an MCMC on function space.
- Immediate problem:
 - if we use a standard random walk, MALA, etc, when working on the infinite space $\beta = \{\beta^j\}_{j=1}^\infty$ the acceptance probability will be zero
 - we know that to have any probability of acceptance, we need to simultaneously propose all of the infinite number of β well.

MCMC on function spaces

- Recall the acceptance probability of Metropolis-Hastings

$$\min \left\{ 1, \frac{l(y | \beta^*) p(\beta^*) q(\beta | \beta^*)}{l(y|\beta) p(\beta) q(\beta^* | \beta)} \right\},$$

which in the case of a random walk is

$$\min \left\{ 1, \frac{l(y | \beta^*) p(\beta^*)}{l(y|\beta) p(\beta)} \right\}.$$

- Which of these terms is making the acceptance probability zero?

Why is the acceptance probability zero?

- $l(y|\beta)p(\beta)$ will not be zero (or we would not be at the current point).
- Is $l(y | \beta^*)$ zero?
 - not unless we have an unusual likelihood where this is possible
 - recall the likelihood is measuring the fit of the n data points under the choice of f^* (as given by the parameters β^*).
- It must be that $p(\beta^*)$ is zero
 - this is the infinite dimensional distribution that is causing the problems
 - using random walk move in function space to propose a function results in a function f^* that has zero probability under the Gaussian process prior.

Motivating pCN

- Key principle in constructing a move:
 - if the target distribution were the prior, the proposed move should have acceptance probability 1.
- Recall the acceptance probability

$$\min \left\{ 1, \frac{l(y | \beta^*) p(\beta^*) q(\beta | \beta^*)}{l(y|\beta) p(\beta) q(\beta^* | \beta)} \right\}.$$

- If we have acceptance probability equal to 1 when the target is the prior, this must mean that the acceptance probability when the target is the posterior will be

$$\min \left\{ 1, \frac{l(y | \beta^*)}{l(y|\beta)} \right\}.$$

- One way to achieve this is to choose the proposal to be the prior

$$q(\beta^* | \beta) = p(\beta^*).$$

Preconditioned Crank–Nicolson (pCN)

- There are other choices that also follow this principle.
- They are based on particular discretisations of an SDE similar to the one we used for MALA
 - the idea is to use discretisations that leave either the prior (or the posterior) invariant.
- One is the *preconditioned Crank–Nicolson* proposal. For any $\alpha \in [0, 1]$, use

$$\beta^* = \sqrt{1 - \alpha^2} \beta + \alpha w,$$

where w is simulated from the Gaussian process prior

- when $\alpha = 1$, this corresponds to drawing from the prior.

Preconditioned Crank–Nicolson (pCN)

- Note that this looks like a bit like an AR-1 process.
- This is derived through a discretisation of a stochastic partial differential equations (SPDE) that has the correct invariant distribution
 - this general case can be used to derive the standard random walk and Langevin proposals
 - it can also be used to derive a preconditioned Crank–Nicolson Langevin proposal for use on function space.
- Details in Cotter, et al. (2013) [MCMC methods for functions: modifying old algorithms to make them faster.](#) Statistical Science 28, 424–446.

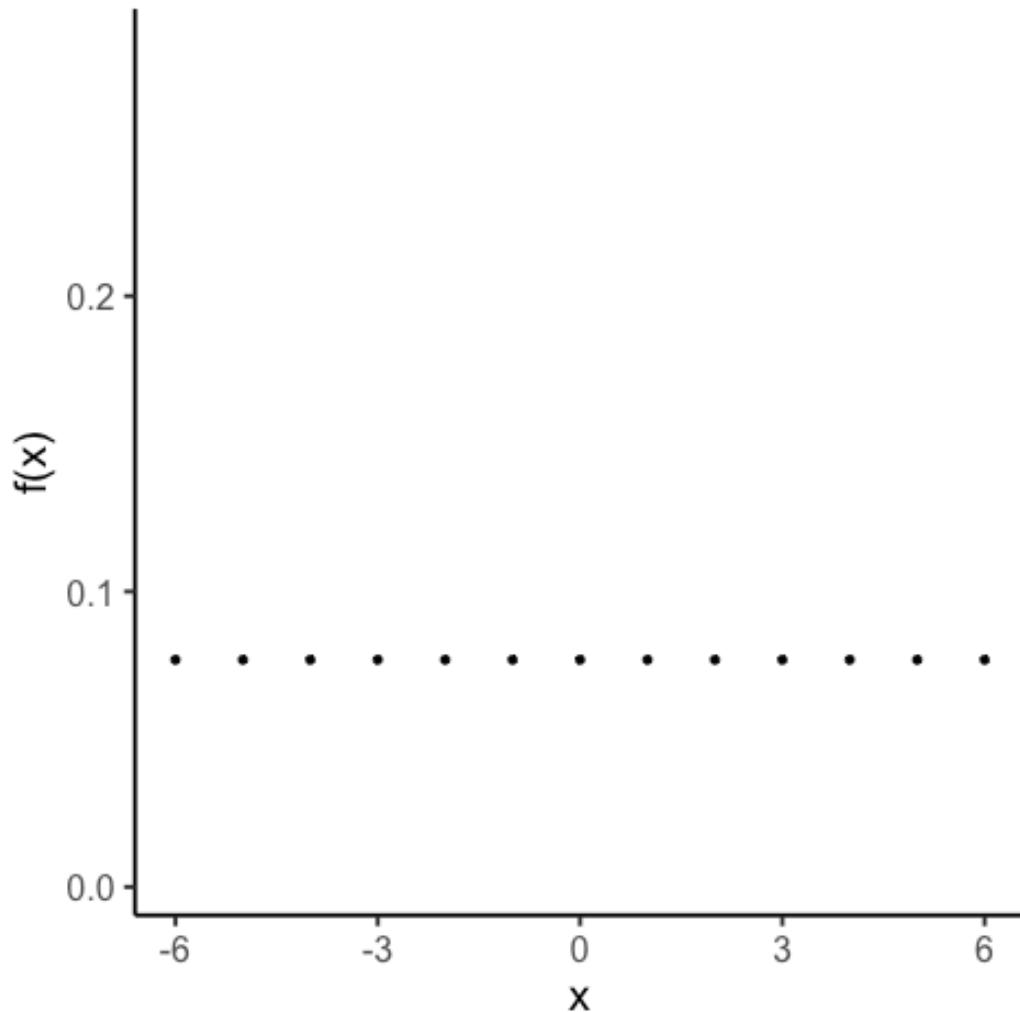
Summary

- Big data is challenging for Bayesian statistics
 - an important reason is that we need to use Monte Carlo methods for inference
 - designing an MCMC method that scales well with p is challenging.
- We have now seen a number of Monte Carlo methods
 - random walk Metropolis $O(p^2)$
 - MALA $O(p^{4/3})$
 - HMC $O(p^{4/3})$.

Summary

- In this lecture we looked at a case where the space is infinite, since it represents a function
 - when implemented in practice, we will use a finite number p of coefficients
 - for most MCMC methods the acceptance probability would go to zero due to the increase in dimension
 - pCN is an MCMC constructed to work (have reasonable acceptance rates) on the infinite space, so when used on a finite representation of the function, it inherits these nice properties.

Example of pCN in action (few points)



Example of pCN in action (many points)

