Advanced Bayesian Computing Introduction

AMS 268

January 4, 2016

1 Parts of the course

1.1 High dimensional regression

This mostly comes from statistical research. This will be taught for roughly 5 weeks.

1.2 Modeling for big data

This more comes from machine learning, but there is still some statistical literature. This will be taught for roughly 3 weeks.

1.3 Approximate Bayes methods

This is almost entirely from machine learning. This will be taught for about 1 week.

2 High dimensional regression

This is when we want to model a response to a large number of predictors. It looks like Raj will represent this as $(y_i, x_{1i}, x_{2i}, \ldots, x_{pi})_{i=1}^n$.

2.1 Penalized optimization

We'll spend some time on **penalized optimization**, which is when we obtain β such that

$$\arg\min_{\beta} ||\mathbf{Y} - \mathbf{X}\beta||^2 + \lambda\Psi(\beta)$$

Where Ψ is a (typically convex) penalty function.

$$\begin{aligned}
\Psi(\beta) &= \textstyle\sum_{i=1}^p |\beta_i| && \text{LASSO} \\
\Psi(\beta) &= \textstyle\sum_{i=1}^p |\beta_i|^2 && \text{Ridge Regression} \\
\Psi(\beta) &= \textstyle\sum_{i=1}^p \left[\frac{\lambda_1}{\lambda_1+\lambda_2}|\beta_i| + \frac{\lambda_2}{\lambda_1+\lambda_2}|\beta_i|^2\right] && \text{Elastic net} \\
\Psi(\beta) &= \textstyle\sum_{i=1}^p \omega_i|\beta_i| && \text{Adaptive LASSO}
\end{aligned}$$

2.2 Model selection

When frequentists talk about high-dimensional regression, it's usually as a **model selection** problem. Basically, if you have p different covariates, then you have 2^p possible models. The mission is to select the best of these models. When Bayesians looked at this, they started with the **g-prior**, and later moved to the **spike-and-slab** prior ($\beta_i \sim \pi\delta_0 + (1-\pi)g$). The spike-and-slab priors in particular suffer heavily when predictors are correlated. There are severe mixing issues, meaning you'll need to run tens of thousands of iterations to converge your MCMC chain, if you're lucky.
One may notice that, in this model selection process, we're testing to see if β_i is exactly equal to zero.

However, this is true with probability zero, so what is currently being worked on is the concept of **shrinkage priors**, which tug very small coefficient values to a value of zero. This keeps important coefficients as nonzero, but makes all relatively unimportant predictors have a coefficient of zero. Shrinkage priors make model selection easy. Another advantage is that shrinkage priors are computationally efficient and scalable up to $p = 5,000$ to 10,000. Raj has a paper that even takes p up to around 50,000 predictors. Shrinkage priors also mix very quickly.
There have been several papers written about Bayesian penalty regression. Bayesian LASSO is not really a good idea. Many other ideas have been published, but now Raj believes that the **Generalized Double Pareto** prior is the current state-of-the-art. Something that is useful in the judgment of these methods is the idea of **posterior consistency**.

3 Modeling big data

This is a situation in which the goal is good prediction. We don't care how many predictors there are, as long as the predictions that the model is churning out are accurate. Since we have a ton of data, we are not as worried about overfitting models, as the sheer amount of data should fix this. Something that is frequently in big data is that we subset the huge dataset, then we use subsequent subsets in the partition of the data to update the results that we obtained in the last subset. The key is to subset and update in a very principled way.

3.1 Issues

1. Storage
2. Computation

3.2 Methods

1. Sequential Monte Carlo
   Draw samples from the prior, then update the estimates as you receive data. The advantage is that you don't have to store all of the data in this case. You can just carry over your parameter estimates.
2. Assumed density filtering
   The posterior is complicated, so it's hard to draw from it directly. What you do instead is you *approximate* the posterior with a parametric distribution.
3. Stochastic gradient descent

   $$\sum_{i=1}^{n} \ell(y_i, \mathbf{x}_i, \boldsymbol{\beta}) = 0$$

   This can be approximated using

   $$\frac{n}{N} \sum_{j=1}^{N} \ell(y_{ij}, \mathbf{x}_{ij}, \boldsymbol{\beta}) = 0$$

   Basically, you make a posterior distribution for each subset in a partition, then you combine them (not simply, but through some methods). This is called **divide and conquer**.

4 Approximate Bayes

4.1 Variational Inference

This idea comes from physics. The idea is that, if you have a complex posterior distribution , you can approximate it with a simpler form. You essentially assume posterior independence.

$$\pi(\theta|\mathbf{y}) \approx q_1(\theta_1|\mathbf{y}) q_2(\theta_2|\mathbf{y}), \ldots, q_p(\theta_p|\mathbf{y})$$

We find these q by minimizing the Kullbeck-Liebler divergence.

4.1.1 Issues

1. Ignoring posterior correlation between parameters, which means you underestimate posterior uncertainty.
2. There's some identifiability issue based on q

4.1.2 Advantages

Variational Bayes is blazingly fast and gives great point estimates.

4.1.3 Application

Variational Bayes is frequently used in NP Bayes models because it drastically reduces the computational load.

4.2 Stochastic Variational Inference

4.3 Streaming Variational Inference

Regression Analysis

AMS 268

January 6, 2016

1 History

- Francis Galton in the 1800s.
- Used to be that fewer than 15 observations was small, 15-50 was unknown, and over 50 was **big data**.
- Today, we'll talk about what happens when we have many predictors for each individuals, but perhaps a small overall sample size.

A motivating example about DNA damage and repair was given. Another example about functional magnetic resonance imaging (fMRI) was given.

2 Formulation

We typically formulate our linear regression as

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p + \epsilon,\ \epsilon \sim \mathrm{N}(0, \sigma^2)$$

where different structures of ϵ can be accommodated. We then try to minimize the squared error to estimate all β_i.

2.1 Error

The **sum of squared error** is a representation of the the overall error in ordinary least squares (OLS) regression. The **sum of squared prediction error** is the SSE plus any bias. The trick here is that **bias** and **error** are inversely related. We want to predict well without overfitting. To do this, we typically try to minimize the **mean squared error** (MSE).
The **Gauss Markov** theorem states that the OLS has the smallest error of all linear unbiased estimators. There could be a biased estimator that is able to attain a lower MSE.
Think about the following:

$$\begin{aligned} C\mathbf{y} &= \left[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}' + D\right] y \\ \mathrm{E}[C\mathbf{y}] &= \boldsymbol{\beta} \\ \mathrm{E}\left[(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}' + D\right] y &= \boldsymbol{\beta} \\ (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{X}\boldsymbol{\beta} + D\mathbf{X}\boldsymbol{\beta} &= \boldsymbol{\beta},\ \forall \boldsymbol{\beta} \\ D\mathbf{X} &= \mathbf{0} \end{aligned}$$

3 Shrinkage Estimation

If the OLS estimate is $\hat{\beta}_j$, what happens to the MSE if we use $\tilde{\beta}_j = \frac{\hat{\beta}_j}{1+\lambda}$? Does the MSE go down? The answer, fo course, is yes. In fact, the value of λ that minimizes the MSE is

$$\lambda = \frac{p\sigma^2}{\sum_{j=1}^{p} \beta_j^2}.$$

To show this, let's think about **orthogonal regression** (i.e. $\mathbf{X}'\mathbf{X} = I$). So

$$\begin{aligned} \mathrm{E}[(\tilde{\beta}_j - \beta_j)^2] = &\quad \mathrm{Var}(\tilde{\beta}_j) + \mathrm{Bias}^2(\tilde{\beta}_j) \\ = &\quad \mathrm{Var}(\tfrac{\hat{\beta}_j}{1+\lambda}) + (\mathrm{E}[\tfrac{\hat{\beta}_j}{1+\lambda}] - \beta_j)^2 \\ = &\quad \tfrac{\sigma^2}{(1+\lambda)^2} + \tfrac{\lambda^2}{(1+\lambda)^2}\beta_j^2 \\ \rightarrow \mathrm{MSE}(\tilde{\beta}) = &\quad \tfrac{p\sigma^2}{(1+\lambda)^2} + \tfrac{\lambda^2}{(1+\lambda)^2}\sum_{i=1}^{p}\beta_j^2 \end{aligned}$$

By taking the derivative, we find that the MSE is minimized when

$$\lambda = \frac{p\sigma^2}{\sum_{j=1}^{p}\beta_j^2}$$

Since we can minimize the MSE in this way, it motivates us to examine shrinkage estimators. Note that when λ gets very large, the estimator approaches zero (λ is a tuning parameter). Charles Stein and his student James found that the James-Stein estimator has lower MSE when σ^2 is known.. Stanley Sclove improved the J-S estimator slightly. If σ^2 is unknown, he proposed n estimator using $c \times$ RSS in place of σ^2, where RSS is the **residual sum of squares**. All of these estimators can be found with a quick Google search.

Now, we often use the F-statistic to evaluate the significance of a regression model. An interesting property of the Sclove estimator is that if $F \times p < c$, then all of the estimators ($\hat{\beta}_j$) are set to zero. However, this is an all-or-nothing shrinkage estimator. This estimator will either decide that all of the estimators are zero, or that none of them are. This is not optimal, so more work must be done.

The next idea to be used is **stepwise regression** in which covariates are either introduced or removed, and then the model is evaluated using AIC, BIC, etc. However, this is not a process that can be automated, so it may take a while to converge upon what you may call the final model, which minimizes MSE. So how can we use the shrinkage parameter to select variables?

4 Ridge Regression

Ridge regression is one possible answer, though it was not developed from this perspective. Remember that for Ridge regression,

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}'\mathbf{y}$$

Regression

AMS 268

January 8, 2016

We have two goals in regression

1. Prediction
2. Variable selection

Last time, we talked about how a slightly biased estimator can result in a lower MSE. This brought us to shrinkage estimation. We ended with an introduction to ridge regression.

1 Ridge Regression

When $n < p$, the matrix $(\mathbf{X}'\mathbf{X})$ is not invertible. One solution is to solve $(\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})\boldsymbol{\beta} = \mathbf{X}'\mathbf{y}$, with small λ. We know that we typically find coefficients by solving

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||^2$$

Does ridge regression lead to an objective solution? Yes. the objective function is

$$\ell(\boldsymbol{\beta}) = ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||^2 + \lambda||\boldsymbol{\beta}||^2$$

By setting $\frac{\delta\ell}{\delta\boldsymbol{\beta}} = 0$, we get the objective function $(\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})\boldsymbol{\beta} = \mathbf{X}'\mathbf{y}$. Raj showed an example in `R` of an advantage to ridge regression over OLS. With highly correlated predictors, the OLS estimation can result in vastly different coefficient values than what the data generating distribution was using. This led us to another question: how should we choose λ?

In general, we use cross validation to choose λ, such as **k fold** cross validation or **leave one out** cross validation.

1.1 k fold cross validation

You partition the data into k subsets, and you'll choose the tuning parameter grid $\lambda \in \{\lambda_1, \lambda_2, \ldots, \lambda_s\}$ where s is the size of the grid of lambda values. Basically, the steps to k fold cross validation are

1. Fit ridge regression model with all but the i^{th} subset of the data, for all values of λ from the grid.
2. Find the average mean squared error for each λ value.
3. Fit the model to the i^{th} subset and choose the value of λ that minimizes the mean squared error.

In other words, let MSE_i be the MSE obtained by fitting the model to every fold other than the i^{th} fold. Then

$$MSE = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

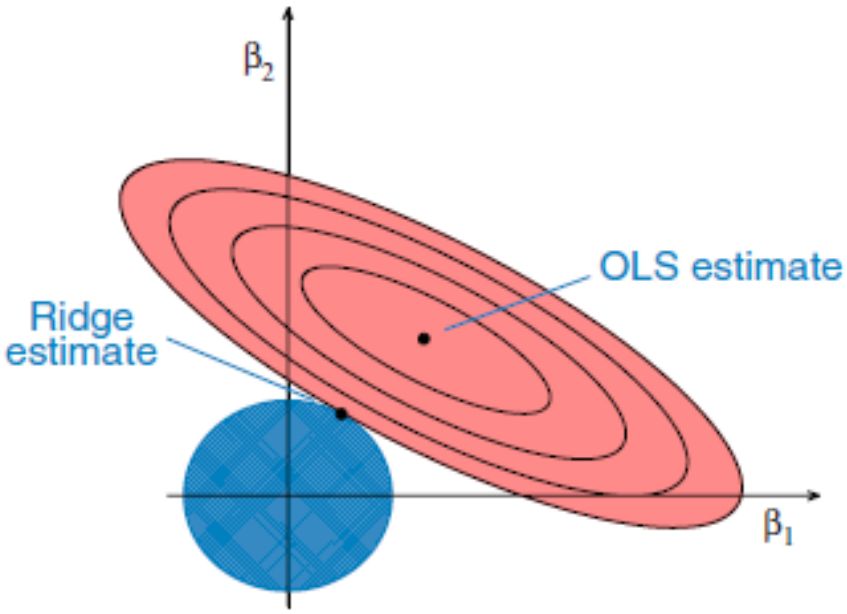


Figure 1: A common visualization of ridge regression

1.2 leave one out cross validation

This is essentially k fold cross validation when $k = n$.

1.3 More on ridge regression

- coefficients decrease in size
- the intercept is not penalized
- coefficients are scale invariant
- ridge regression is not suitable when one or two coefficients are large and everything else is zero (this is more suited to LASSO)
- ridge regression is better when there are several nonzero coefficients that are not very large
- one criticism is that the sequence of λ that is chosen is very important
- shrunken coefficient values are zero only when $\lambda = \infty$
- ridge regression **cannot select variables**

Note: In `mass` in R, ridge regression performs LOOCV, and LASSO does 10-fold CV.

1.4 How OLS finds $\boldsymbol{\beta}$

We use $||\mathbf{y} = \mathbf{X}\boldsymbol{\beta}||^2$ to find $\boldsymbol{\beta}$, and

$$||\mathbf{y} = \mathbf{X}\boldsymbol{\beta}||^2 = (\boldsymbol{\beta} - \hat{\boldsymbol{\beta}})'\mathbf{X}'\mathbf{X}(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}) + \mathbf{y}'[\mathbf{I} - P_x]\mathbf{y}$$

Ridge regression basically sets a constraint such that $||\boldsymbol{\beta}||^2 < \lambda$. A visualization of ridge regression can be seen in figure 1.

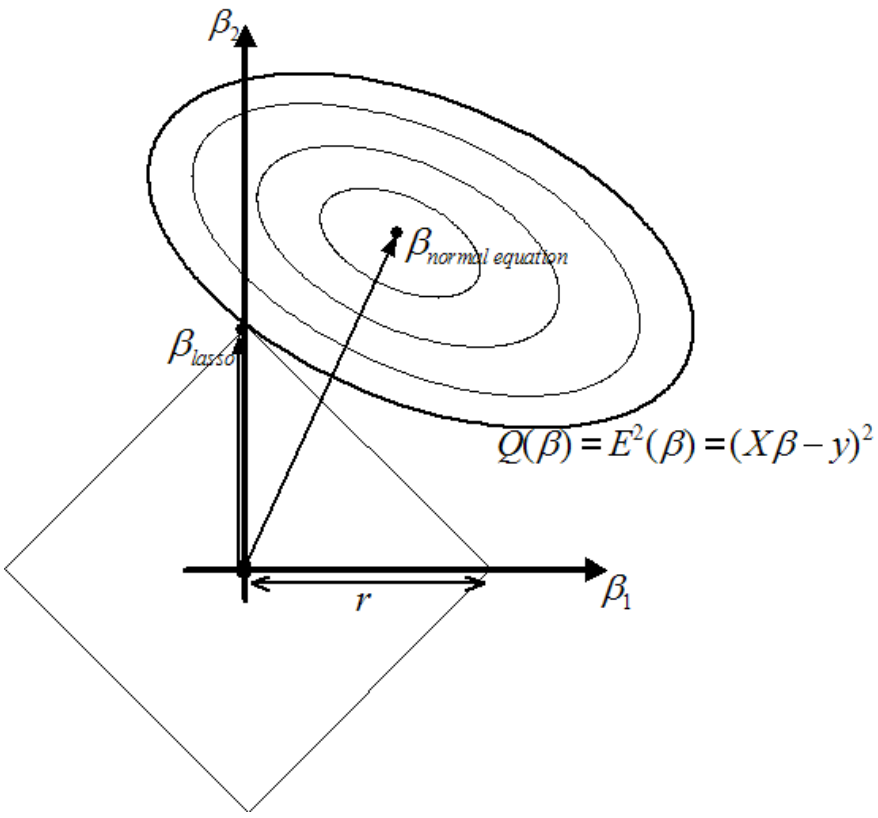


Figure 2: A common visualization of LASSO

1.5 Postprocessing

We could, if we wanted to do some variable selection, make a confidence interval for each coefficient and discard all coefficients which have confidence intervals containing zero.

2 LASSO

LASSO combines the shrinkage of ridge regression with variable selection. Now the function used to obtain coefficients with ridge regression is

$$\arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} (y_i - \mathbf{x}_i'\boldsymbol{\beta})^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

So now the penalty function is such that $\sum_{j=1}^{p} |\beta_j| \leq \lambda$. A visualization of LASSO can be seen in figure 2. Note that the acceptable region in LASSO resembles a diamond rather than a circle.

Regularized Regression

AMS 268

January 11, 2016

Last time, we talked about ridge regression, and began talking about LASSO.

1 LASSO

The term LASSO means **least absolute shrinkage and selection operator**. Note: You cannot use this penalty:

$$||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||^2 + \lambda||\boldsymbol{\beta}||_0$$

where $||\boldsymbol{\beta}||_0$ is the **sparsity** of $\boldsymbol{\beta}$. That penalty is not *convex*, so there is no way to optimize. What we need to use is

$$||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||^2 + \lambda\Psi(\boldsymbol{\beta})$$

where Ψ is a convex function. So, we say that LASSO uses the penalty

$$||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||^2 + \lambda||\boldsymbol{\beta}||_1$$

As λ increases, the number of predictors used decreases, so the idea is to have the proper value of λ so that you get a good model fit, as well as accurate variable selection. As with ridge regression, we find the proper value of λ using generalized cross validation. Once λ is fixed, what we need to solve is

$$\arg\max_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||^2 + \lambda||\boldsymbol{\beta}||_1^2$$

This finds us $\boldsymbol{\beta}_{LASSO}$. Note that the LASSO penalty is not strictly convex, but merely convex (see below). By contrast, ridge regression does have a strictly convex penalty.

1.1 LARS

In 2004, Hastie, Tibshiriani, Johnstone, and Friedman did this using the LARS algorithm, and called it **least angle regression**. The advantages of LARS are as follows:

1. Computationally as fast as the forward-backward (stepwise) selection algorithm.
2. The angle between a predictor and the error is calculated. If two predictors have approximately the same angle to the error, then you increase or decrease the next coefficient estimate by approximately the same amount.

If, for example, $x_i = x_j$, and we're optimizing

$$||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||^2 + \lambda J(\boldsymbol{\beta})$$

Such that $J(\boldsymbol{\beta})$ is a **strictly convex** function. A function is strictly convex if

$$J(\alpha_1\gamma + \alpha_2(1-\gamma)) < J(\alpha_1)\gamma + (1-\gamma)J(\alpha_2).$$

Then $\hat{\beta}_i = \hat{\beta}_j$. The LARS algorithm can be implemented using the `lars` package in R.

1.2 Coordinate Descent Algorithm

Say the objective function is

$$F(\boldsymbol{\beta}) = ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||^2 + \lambda\Psi(\boldsymbol{\beta})$$

where Ψ is a convex function. Now we want to find

$$\min_{\boldsymbol{\beta}} F(\boldsymbol{\beta})$$

where you start with initial values for $\boldsymbol{\beta}$.

$$\boldsymbol{\beta}^0 = \left(\beta_1^0, \ldots, \beta_p^0\right)$$

The algorithm works as follows:

1. $$\min_{\beta_1} F(\beta_1, \beta_2^0, \ldots, \beta_p^0) \to \beta_1'$$
2. $$\min_{\beta_2} F(\beta_1', \beta_2, \beta_3^0, \ldots, \beta_p^0) \to \beta_2'$$
3. and so forth until
4. $$\min_{\beta_p} F(\beta_1', \beta_2', \beta_3', \ldots, \beta_{p-1}', \beta_p) \to \beta_p'$$
5. Then you perform this over and over until you reach the global solution.

This can be implemented using the `glmnet` package in R. This has an advantage over LARS in that it's much more stable when you have $n \ll p$. Raj gave a code example in class (which he promises to post online today).

2 Elastic Net

Variable selection is constrained by the sample size. For example, if $n = 100$ and $p = 2000$, then LASSO can only make 100 variables nonzero. LASSO also fails to perform group variable selection (if a groups of variables are highly correlated, LASSO tends to select only one of them). This motivates the idea of the elastic net. You are throwing a net to catch multiple fish. Elastic net basically forms a hybrid between LASSO (L_1) and ridge regression (L_2). The penalty becomes

$$J(\boldsymbol{\beta}) = \lambda_1||\boldsymbol{\beta}||^2 + \lambda_2||\boldsymbol{\beta}||_1$$

The L_1 part of the penalty generates a sparse model and the quadratic part (L_2) removes the limitation on the number of selected variables, and encourages the grouping effect. In fact these are our two goals:

1. Enable group variable selection
2. Select more preditors than the sample size

The fact that elastic net accomplishes this first goal is answered by a previous theorem after noting the fact that the elastic net penalty is a strictly convex function. The second can be shown by rewriting the elastic net optimization function

$$|| \begin{pmatrix} \mathbf{y} \\ 0 \end{pmatrix} - \frac{1}{(1+\lambda_2)^{1/2}} \begin{pmatrix} \mathbf{X} \\ \sqrt{\lambda_2}\mathbf{I} \end{pmatrix} \boldsymbol{\beta}^*||^2 + \gamma||\boldsymbol{\beta}^*||_1$$

where $\boldsymbol{\beta}^* = \sqrt{1+\lambda_2}\boldsymbol{\beta}$, and $\gamma = \frac{\lambda_2}{\sqrt{1+\lambda_2}}$. Since LARS and glmnet can select the number of variables equal to, at most, the sample size, at most (n+p) variables can be selected. Since $(n+p) > p$, potentially all variables can be selected. The solution to the optimization problem

$$||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||^2 + \lambda_1||\boldsymbol{\beta}||_1 + \lambda_2||\boldsymbol{\beta}||_2$$

is known as the **naive elastic net estimate**.

Tweaking LASSO and Intro to High-Dimensional Clustering

AMS 268

January 13,2016

Last class, we talked about LASSO (and two algorithms to perform LASSO, LARS and coordinate descent). We also began talking about the elastic net regularization method, which is a hybrid of LASSO and ridge regression. Elastic net brings the best of LASSO and ridge regression together because it performs variable selection, while allowing for grouped predictors and allowing for a number of nonzero coefficients that are greater than n.

1 Naive Elastic Net

We ended last lecture by discussing the **naive elastic net estimate**.

$$\hat{\boldsymbol{\beta}}_{elastic} = \arg\min ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||^2 + \lambda_1||\boldsymbol{\beta}||_1 + \lambda_2||\boldsymbol{\beta}||_2$$

In the notation that we used last time, now

$$J(\boldsymbol{\beta}) = \lambda_1||\boldsymbol{\beta}||^2 + \lambda_2||\boldsymbol{\beta}||_1$$
$$\rightarrow Q(\boldsymbol{\beta}) = \frac{\lambda_1}{\lambda_1+\lambda_2}||\boldsymbol{\beta}||^2 + \frac{\lambda_2}{\lambda_1+\lambda_2}||\boldsymbol{\beta}||_1 = \alpha||\boldsymbol{\beta}||^2 + (1-\alpha)||\boldsymbol{\beta}||_1$$

However, this produces a slightly biased estimator due to the double penalization. We then adjust the estimate to find the **actual elastic net** estimator.

$$\boldsymbol{\beta}_{enet} = (1+\lambda_2)\boldsymbol{\beta}_{naive\,enet}$$

2 Adaptive LASSO

Some research has been done to improve upon LASSO. Remember, the penalty in LASSO is

$$J(\boldsymbol{\beta}) = ||\boldsymbol{\beta}||_1 = \sum_{j=1}^{p} |\beta_j|$$

This has been changed slightly to become **adaptive LASSO**.

$$J(\boldsymbol{\beta}) = \sum_{j=1}^{p} \omega_j|\beta_j|$$

where

$$\omega_j = \frac{1}{\hat{\beta}_J^{OLS}}.$$

This is an improvement over LASSO because it yields consistent estimates of the parameters while alsoretaining the attractive properties of LASSO (variable selection). Another idea has been to run **individual regressions** for each predictor. Consider the predictors $x_1, \ldots, x_p$. Then one could fit

$$y = \beta_0 + \beta_j x_j + \epsilon_j,\ \epsilon \sim \mathrm{N}(0, \sigma^2)$$

for $j = 1, \ldots, p$. This has become a very popular practice. To reiterate,

Even when we run LASSO, a popular practice is to run p independent regressions of the type

$$y = \beta_0 + \beta_1 x_j + \epsilon,\ \epsilon \sim \mathrm{N}(0, \sigma^2)$$

with the regression coefficients, run p independent t-tests to infer on the hypothesis

$$H_{0j} : \beta_j = 0 \,\mathrm{vs}\, H_{1j} : \beta_j \neq 0,\ j = 1, \ldots, p$$

This *prescreening* reduces numerical instability, and makes the estimates more consistent. If some predictors are very noisy (not contributing much to the regression), AL (adaptive LASSO) will penalize them more heavily that LASSO. Fortunately for us, running adaptive LASSO is easy and converges to the true parameter values (proved by Zuo in 2007). The way you can run it is as follows. Consider the design matrix $\mathbf{X} = [\mathbf{x}_1 : \mathbf{x}_2 : \ldots : \mathbf{x}_p]$. You change the design matrix to

$$\mathbf{X}^{**} = \left[\frac{\mathbf{x}_1}{|\hat{\boldsymbol{\beta}}_1^{OLS}|^\nu} : \frac{\mathbf{x}_2}{|\hat{\boldsymbol{\beta}}_1^{OLS}|^\nu} : \ldots : \frac{\mathbf{x}_p}{|\hat{\boldsymbol{\beta}}_1^{OLS}|^\nu} \right]$$

Then you run regular old LASSO with

$$\arg\min_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}^{**}\boldsymbol{\beta}|| + \lambda ||\boldsymbol{\beta}||_1$$

because $\hat{\boldsymbol{\beta}}_{LASSO} \rightarrow \hat{\boldsymbol{\beta}}_{AL} = \frac{\hat{\boldsymbol{\beta}}_{j,LASSO}}{|\hat{\boldsymbol{\beta}}_j^{OLS}|^\nu}$. There is a bounded region for ν, though a value of 2 is typically used. Check out a lecture on ADMM by Boyd in 2011. There is also literature on **Factor machines** which includes a discussion on interactions. The adaptive LASSO algorithm can be run using the `polywog` package in R.

3 Group LASSO

In some settings, certain predictors belong to predefined groups. In these cases, it's desirable to shrink and select members of these groups together. **Group LASSO** is one way to achieve this. Suppose that the p predictors are divided into m groups with p_j as the number of predictors in group j, such that $p_1 + p_2 + \ldots + p_m = p$. Group LASSO minimizes

$$\arg\min_{\boldsymbol{\beta} \in \mathbb{R}^p} \left[||\mathbf{y} - \beta_0 \mathbf{1} - \sum_{j=1}^{m} \mathbf{X}_j \boldsymbol{\beta}_j||^2 + \lambda \sum_{j=1}^{m} \sqrt{p_j} ||\boldsymbol{\beta}_j||_2 \right]$$

where $\mathbf{X}_j$ is the matrix corresponding to the j^{th} group of predictors and $\boldsymbol{\beta}_j$ is the vector corresponding to $\mathbf{X}_j$.

3.1 Oracle property

If we're in a regression setting

$$y = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p + \epsilon$$

then we call $\mathcal{A}$ the number of nonzero coefficients in the true model, and $\mathcal{A}_n$ is our support for the number of nonzero coefficients. The **oracle property** states that with our penalized optimization scheme, we identify $\mathcal{A}_n$ such that

$$P(\mathcal{A}_n = \mathcal{A}) \xrightarrow{\text{as } n \to \infty} 1$$

4 Clustering in high dimensions: nonnegative matrix factorization

This is something frequently done as a first step by data miners. Let's say we're in the genetic modeling setting. Thus, we have a response y and many, many predictors $(\mathbf{x}_1, \ldots, \mathbf{x}_p)$ where p is huge, say 2000. Additionally, in this setting, each preditor can take only one of three values: 0, 1, and 2. (This is the structure of SNPs (Single Nucleotide Polymorphism)). Say you want to cluster different subjects in the sample into k clusters. Many people try k-means clustering, which Raj says is shit in high dimensions. We'll talk more about this next time.

Clustering in High Dimensions and Penalized Optimization

AMS 268

January 15, 2016

1 High-Dimensional Clustering

One reason why clustering in high dimensions is needed is because sometimes there are differences in large groups of predictors in a data set. Raj gave an example on genetics and race. What happens is you divide your sample into k clusters. This is analogous to k-means clustering, but k-means clustering is based on distance. However, in high dimensions, the distance between sets of two high-dimensional observations is very similar, so basing the clustering on distance is not a good way to go. The methods we can use instead are:

1. Spectral clustering
2. Clustering with non-negative matrix factorization (NMF)

When NMF was designed, it was not meant for clustering.

1.1 Definition

Say we have the transpose of the predictor matrix $\mathbf{M}_{p\times n}$. We want to find two matrices, $\mathbf{W}_{p\times k} > 0$ and $\mathbf{H}_{k\times n} > 0$, that is, every element in each matrix is greater than 0, such that $\mathbf{M} \approx \mathbf{WH}$. This is found by solving te optimization problem

$$\min_{\mathbf{W}>0,\mathbf{H}>0} ||\mathbf{M} - \mathbf{WH}||^2$$

Why do this when SVD (singular value decomposition) does a better job of approximating $\mathbf{M}$? The SVD is done as

$$\mathbf{M} = \mathbf{U}\Lambda\mathbf{V}.$$

So, if we subset the SVD appropriately,

$$||\mathbf{M} - \mathbf{U}_k\Lambda_k\mathbf{V}_k||^2 \leq ||\mathbf{M} - \mathbf{WH}||^2$$

However, for non-negative data, the NMF approximation provides a much better interpretation.

1.2 NMF and K-Means Clustering

One could write k-means clustering as $||\mathbf{M} - \mathbf{WH}||^2$. The way k-means clustering is typically written as

$$\arg\min_{\mathbf{w}_i} \sum_{i=1}^{k}\sum_{j=1}^{n_i} ||\mathbf{x}_{ij} - \mathbf{w}_i||^2$$

which could be written as

$$\arg\min_{\mathbf{W}} ||\mathbf{M}_{p\times n} - \mathbf{W}_{p\times k}\mathbf{H}_{k\times n}||^2.$$

In this scenario, the columns of $\mathbf{W}$ are the cluster centers and the columns of $\mathbf{H}$ are the cluster membership indicators. SO, to reiterate, the objective function in the k-means clustering can be written as $||\mathbf{M} - \mathbf{WH}||^2$ where $\mathbf{W}_{p\times k}$ is a matrix whose i^{th} column corresponds to the i^{th} cluster center. $\mathbf{H}$ is a $k \times n$ matrix where the j^{th} column appears as $(0, \ldots, 0, \underbrace{1}_{\ell}, 0, \ldots, 0)$, where 1 is in the ℓ^{th} element, signifying that the j^{th} sample belongs to the ℓ^{th} cluster.

Note: Raj introduced this such that all elements of $\mathbf{W}$ and $\mathbf{H}$ are positive, but, as the name implies, the Wikipedia page [1] states that the constraint is that all elements of $\mathbf{W}$ and $\mathbf{H}$ are **nonnegative**. I think he was just thinking of genomic data, where the data matrices are all positive.

1.3 Sparse NMF

This is a variation of NMF where the objective function is

$$\arg \min_{\mathbf{W},\mathbf{H}>0} ||\mathbf{M} - \mathbf{WH}||^2 + \lambda \sum_{j=1}^{n} ||\mathbf{H}(,j)||_1$$

1.4 Nonlinear Data

This method works well when there is a highly non-linear data generating process. The idea is to split into many, many clusters and then fit a standard regression to each cluster.

2 Penalized Optimization

Penalized optimization is unable to provide satisfactory predictive inference: it only provides point predictions. However, in many scientific applications, some uncertainty characterization is desired. Another issue is that tuning parameters can affect the inference greatly.

2.1 Bayesian Approach

We can certainly make this better, though! If the loss function corresponds to a likelihood and penalty to the log pprior (up to normalizing constants), then estimates correspond to the MAP estimate. However, the MAP is not the Bayes estimator for a reasonable loss function. Also, we want the whole posterior for inference, rather than a simple point estimate.

[1] `https://en.wikipedia.org/wiki/Non-negative_matrix_factorization`

Spike and Slab Prior and SVSS

AMS 268

January 20, 2016

1 Problems with Penalized Optimization

We talked a little bit last class about the issues with penalized optimization. Mainly, penalized optimization gives only point estimates, with no measure of uncertainty. This leads us to using a bootstrap.

2 Bootstrap to find predictive intervals

Basically, we fit the model multiple times on different subsets of the data, and then construct confidence intervals for each parameter in $\boldsymbol{\beta}$.

Problem: This is not applicable in high dimensions.

3 A Bayesian Approach

We can use the log likelihood and the prior to try to estimate the parameter values. However, this corresponds to the posterior mode, which corresponds to the 0-1 loss function, which is not a good loss function to use in this instance. So what we need to do is to calculate the posterior distribution. When $n \gg p$,

$$\pi(\boldsymbol{\beta}, \sigma^2 | \mathbf{y}, \mathbf{X}) \approx \mathrm{N}(\boldsymbol{\beta} | \hat{\boldsymbol{\beta}}, \mathbf{I}(\boldsymbol{\beta})^{-1}),$$

which is called the **Bernstein-Von Mises Theorem** or the **Bayesian Central Limit Theorem**. What this means, in essence, is that when $n \gg p$, the prior does not play much of a role in determining the posterior. What this means is that the frequentist and Bayesian estimates are essentially equivalent. However, this does not hold when p is large. In this case, the prior will have a profound effect, so it is essential to very carefully construct the prior.

3.1 Prior Design

The priors in this caseneed to be designed such that the posterior of $\boldsymbol{\beta}$ concentrates around the "true" $\boldsymbol{\beta}_0$. The prior needs to be somewhat informative, as flat priors will lead to some inconsistencies in the posterior. One idea is to impose some sparsity on $\boldsymbol{\beta}$ through the prior distributions. Later on in this course, we will see that we may have other things to take into consideration when designing the priors.

3.2 Variable selection via sparsity

3.2.1 Spike and Slab Prior

$$\beta_j \overset{iid}{\sim} \pi_0 \delta_0 + (1 - \pi_0) g$$

One popular selection of g is $\mathrm{N}(0, c)$. In this notation, π_0 is the prior probability of excluding a predictor, and δ_0 is the degenerate distribution.

The indicator of which variables are included in the model can be represented by $\gamma_j = I(\beta_j \neq 0)$. Thus, a candidate model can be represented by $\boldsymbol{\gamma}$, the vector of length p that characterizes whether or not a variable is included. Therefore, $\sum_{j=1}^{p} \gamma_j \sim \text{Binomial}(p, 1 - \pi_0)$. But how can be choose the correct value for π_0? THe obvious answer, of course, is to put a prior on it ($\pi_0 \sim \text{Beta}(\alpha, \beta)$). This is a way for us to get around having to test all of the 2^p models. Still, we need to find the correct model by searching in the correct area of the model space.

3.2.2 Stochastic Search Variable Selection

This is a way for us to find the correct model. We move between models, and return to models that more accurately represent the data. This relies on MCMC to conduct the search. Now, we have

$$\beta_j \sim (1 - \gamma_j)\text{N}(0, v_{0j}) + \gamma_j \text{N}(0, v_{1j}),\ \gamma_j \overset{ind}{\sim} \text{Ber}(w_j)$$

It's best to run this when v_{0j} is small and v_{1j} is "reasonably" big. From here the model implementation is relatively straightforward. Something that Raj suggests that we read is Bayerri (Barbieri?) & Berger's 2004 paper on the Median Probability Model.

3.2.3 Problems with SSVS

The MCMC runs for a long time and hops between models. The posterior probability of the model is estimated by the number of times that the model is visited by the Markov chain. This is problematic when there are high correlations between variables. SSVS is also not helpful if you want to have flat priors on any of the β_js. The choice of g can also be a nightmare. Finally, this method is not typically seen as scalable to really big p, but we can use GPUs and other tricks to help this out. Raj says that this method was great in the 90s, but now we have much larger datasets, and it has become too cumbersome for present-day problems.

Stochastic Search Variable Selection and Continuous Shrinkage Priors

AMS 261

January 22, 2016

1 Stochastic Search Variable Selection

1.1 Bayesian Inference

Now we can quantify uncertainty! We can also estimate **marginal inclusion** probabilities $P(\gamma_j = 1|\mathbf{y}, \mathbf{X})$. This is just the proportion of times that the MCMC visits a model including the j^{th} variable. This gives us a measure of how important a predictor is. Berger (and one of his students) proposed a marginal inclusion cutoff of 0.5. Remember, we fit models sequentially and test them. For example, we start out with predictors $x_1, \ldots, x_p$, and we find that the "best" model has only β_1 and β_3 as nonzero. Then, using SVSS, the final model will look something like

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots + \beta_p x_p + \epsilon$$

Where all of the variables are included, but only β_1 and β_3 are far from zero.

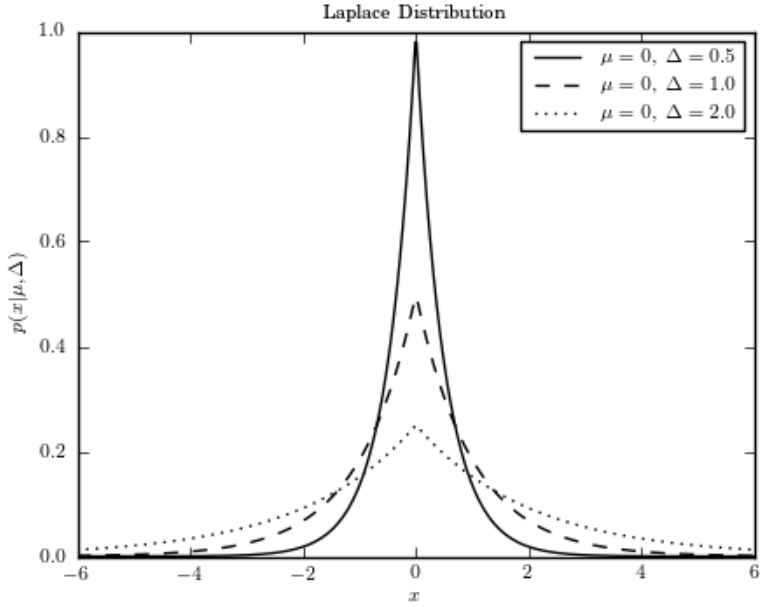
Some people do what is known as **model averaged prediction**. This is when you perform predictive inference with each model, but then you take the weighted average to aggregate your results from alll models, using each γ_j as the weight for the prediction of each individual model. If you carry out model averaged prediction, then your final predictive estimates should not correspond to predictive estimates from a model with fewer variables.

1.2 Advantages and Disadvantages

- We can select variables!
- The process is cumbersome for large p.
- Can have some problems with highly correlated predictors.
- Sparsity can be induced in a "weaker" sense. That is, we don't have to say that $\beta_j = 0$ in order to remove it from the model. If $\beta_j \approx 0$, then we can still exclude it. This can be a positive point for Bayesians who are philosophically opposed to hypothesis testing.

2 Continuous Shrinkage Priors

As before, shrinkage priors are continuous prior distributions that pull unimportant predictor coefficients to zero while leaving nonzero coefficients untouched. The idea is to have a prior that looks something like this:



Note that the Laplace (double exponential) distribution is a popular choice as a shrinkage prior.

3 Global Local Representation

In a paper by Polson and Scott in 2010, they show that essentially all shrinkage priors can be represented as

$$\beta_j|\psi_j, \tau \overset{ind}{\sim} \mathrm{N}(0, \psi_j\tau),\ \psi_j \overset{iid}{\sim} g,\ \tau \sim f.$$

The global scale τ facilitates the concentration near zero. The local scales ψ_j are highly variable to prevent overshrinking β_js corresponding to important predictors.This representation as a scale mixture of normals allows for easy Gibbs sampling. In this way, a strong global shrinkage handles the noise in the data, while the local ψ_js find the signals, which would be considered outliers relative to τ. We have already discussed a **spike-and-slab** prior, which is structured as

$$\begin{gathered}\beta_j \sim (1-w)\delta_0 + wg(\beta_j) \\ \mathrm{E}[\beta_j|y, w] \approx w(y_j)y_j\end{gathered}$$

where

$$w(y_j) = \frac{w\int \mathrm{N}(y_j|\beta_j, \sigma^2)g(\beta_j)\pi(\sigma^2)}{(1-w)\int \mathrm{N}(y_j|0, \sigma^2)g(\beta_j)\pi(\sigma^2) + w\int \mathrm{N}(y_j|\beta_j, \sigma^2)g(\beta_j)\pi(\sigma^2)}$$

If we have a lot of noisy predictors, $\pi(w|\mathbf{y})$ will be concentrated around 0, so $w(y_j)$s are pulled to 0 by w in the numerator. However, for more important coefficients, $w(y_j) \approx 1$. This can only be achieved if $\int \mathrm{N}(y_j\beta_j, \sigma^2)g(\beta_j)\pi(\sigma^2)$ outweighs w, so g has to have a heavy tail. In the global-local setting

$$\begin{gathered}\beta_j|\psi_j, \tau \sim \mathrm{N}(0, \psi_j\tau) \\ \psi_j \overset{iid}{\sim} g \\ \tau \sim f \\ \mathrm{E}[\beta_j|\mathbf{y}, \psi_j, \tau] = \left(1 - \tfrac{1}{1+\psi_j\tau}\right) y_j\end{gathered}$$

This clearly shows the effect of the local versus the global scales. The machine learners like to say **shrink globally, act locally**. If there are many noisy predictors, $\pi(\tau|\mathbf{y})$ will be concentrated around zero. Thus,

$\left(1 - \frac{1}{1+\psi_j\tau}\right)$s are pulled to zero. However, ψ_js for important predictors are so large that $\left(1 - \frac{1}{1+\psi_j\tau}\right) \approx 1$. A couple of guidelines:

1. g should be heavy-tailed
2. f should have sufficient mass around zero.

3.1 Early Shrinkage Priors

- A shrinkage prior with the structure:

$$\beta_j|\tau, \psi_j \sim \mathrm{N}(0, \tau^2\psi_j^2),\ \psi_j^2 \sim \mathrm{IG}(\zeta/2, \zeta/2),\ \tau^2 \sim \mathrm{IG}(a, b)$$

  This gives rise to $\beta_j|\tau \sim t_\zeta(0, \tau)$.

- Strawderman-Berger Prior: $\beta_j|\psi_j \sim \mathrm{N}(0, \psi_j^{-1} - 1),\ \psi_j \sim \mathrm{Beta}(1/2, 1)$.
- Bayesian LASSO (Park and Casella, 2008; Hans 2009)

Note: Normal is our baseline for tail size. Distributions with thicker tails that the normal distributions are considered "heavy"-tailed. While the Laplace distribution doesn't have super thick tails, it does better than the normal distribution.

Shrinkage Priors

AMS 268

January 25, 2016

Last time, we discussed the development of the global-local shrinkage priors. They are structured such that

$$\begin{gathered}\beta_j \sim \mathrm{N}(0, \psi_j \tau) \\ \tau \sim f \text{ with lots of mass near } 0 \\ \psi_j \overset{iid}{\sim} g \text{ with a heavy tail}\end{gathered}$$

This is both computationally simple and generally does not have a mixing issue (unlike spike-and-slab priors). Next, we discussed the Bayesian LASSO prior (2008)

$$\pi(\boldsymbol{\beta}|\sigma^2) = \prod_{j=1}^{p} \frac{\lambda}{2\sqrt{\sigma^2}} \exp(-\lambda|\beta_j|/\sigma)$$

This is nice because the negative logarithm produces a frequentist LASSO-like penalty function. For the λ variable, we can assign the prior

$$\pi(\lambda^2) = \frac{\delta^r}{\Gamma(r)}(\lambda^2)^{r-1}e^{-\delta\lambda^2}$$

to avoid having to perform cross-validation to find the optimal value for λ. Note: the double-exponential distribution (for β_j) can be rewritten as a scale mixture of normals.

$$\begin{gathered}\boldsymbol{\beta}|\tau_1^2, \ldots, \tau_p^2 \sim \mathrm{N}(\mathbf{0}, \sigma^2 diag(\tau_1^2, \ldots, \tau_p^2)) \\ \pi(\tau_j^2) = \tfrac{\lambda^2}{2} \exp\left(-\tfrac{\lambda^2 \tau_j^2}{2}\right) \\ \sigma^2 \sim \pi_\sigma\end{gathered}$$

Raj gave the full conditional distributions in his slides (page 45 or 46). One of the distributions that pops out is the inverse-gaussian [1]. Raj also notes that you can use other priors for λ^2. You could also use an empirical Bayes approach.

1 Horseshoe Prior

This is proposed for a normal means problem ($y_p i = \theta_i + \epsilon_i$, $\epsilon_i \sim \mathrm{N}(0, \sigma^2)$). The prior is set up as

$$\begin{gathered}\theta_i|\lambda_i \sim \mathrm{N}(0, \tau^2\lambda_i^2) \\ \lambda_i \sim C^+(0,1) \\ \tau \sim C^+(0,1)\end{gathered}$$

Where C^+ is the *folded Cauchy* or *truncated Cauchy* distribution. If we find the expected value of each coefficient β_i, we'll find that

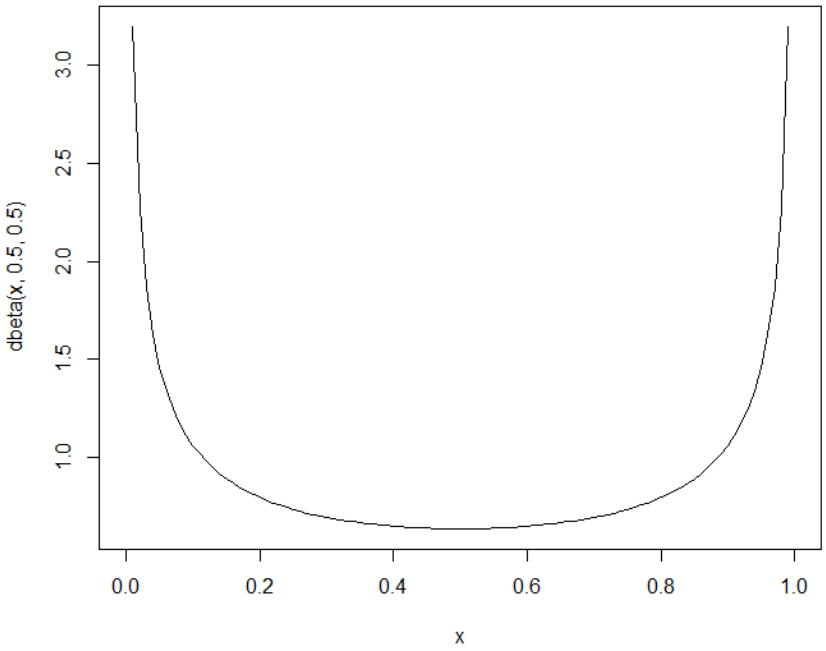
$$\begin{gathered}\mathrm{E}[\theta_i|\mathbf{y}, \lambda_i, \tau] = (1-\kappa_i)y_i \\ \kappa_i = \tfrac{1}{1+\lambda_i^2\tau^2}\end{gathered}$$

[1] `https://en.wikipedia.org/wiki/Inverse_Gaussian_distribution`

So what is the density for κ_i? Raj begins by setting $\tau = 1$. Then

$$\begin{aligned} P(\tfrac{1}{1+\lambda^2} \leq y) = & \quad P\left(\lambda_i^2 \geq \tfrac{1}{y} - 1\right) \\ = & \quad P(\lambda_i \geq \sqrt{1 - \tfrac{1}{y}}) \\ = & \quad \int_{\sqrt{1-\frac{1}{y}}}^{\infty} \frac{2}{\pi(1+x^2)} dx \\ = & \quad \int_0^y \frac{1}{z^{1/2}(1-z)^{1/2}} \frac{1}{\text{Beta}(1/2,1/2)} dz \end{aligned}$$

where $x = \sqrt{1/z - 1}$. This is the CDF for a Beta distribution, so $\kappa_i \sim \text{Beta}(1/2, 1/2)$. This is called a horseshoe prior because the density function for κ_i looks like this:



We do this such that

1. If a θ_i is unimportant, we would like the corresponding κ_i to concentrate around 1.
2. If a θ_i is important, we want κ_i to concentrate around 0.

This can happen easily because this prior puts a large amount of mass at 0 and 1. Raj says this prior does **very** well. In fact, he goes so far as to say that the horseshoe prior is a "nightmare" to beat. Unfortunately, you don't get a closed-form posterior distribution. You will have to develop a M-H algorithm, but this gives rise to choosing the correct tuning parameters. For large p, this is quite cumbersome. The nonconjugate variables will be the λ_i and τ. One thing some do is approximate τ with an inverse gamma. However, the moral of the story is that, for massive p, horseshoe is not recommended. If you'd like to implement this in R, use the package `monomvn`.
Raj showed a handy table that gives the density for λ_i and κ_i for different priors for θ_i.

2 Generalized Double Pareto Prior

This is a prior developed in 2013 by one of Dunson's former postdocs. It is available in *Statistica Sinica*. It looks like

$$\pi(\beta) = \frac{1}{2\xi}\left(1 + \frac{|\beta|}{\alpha\xi}\right)^{-(\alpha+1)}, \ \beta \in \mathbb{R}$$

Shrinkage Priors

AMS 268

January 27, 2016

Last time, we were talking about shrinkage priors, namely the horseshoe prior (for a normal means problem). We began talking about the Generalized Double Pareto Prior when we ran out of time.

1 Generalized Double Pareto Prior

This is a prior developed in 2013 by one of Dunson's former postdocs. It is available in *Statistica Sinica*. It looks like

$$\pi(\beta) = \frac{1}{2\zeta}\left(1 + \frac{|\beta|}{\alpha\zeta}\right)^{-(\alpha+1)}, \ \beta \in \mathbb{R}$$

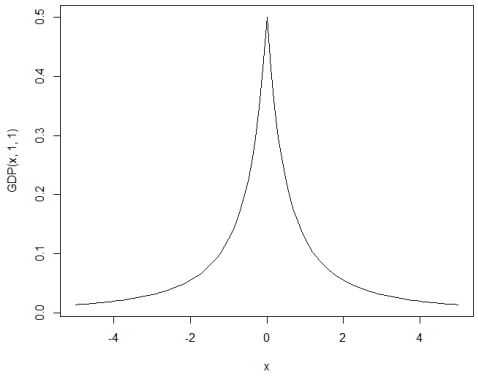
If we were to set $\alpha = \zeta = 1$, then the prior becomes the standard double pareto distribution. If we're set up as

$$\begin{gathered}\beta \sim \mathrm{N}(0, \tau) \\ \tau \sim \text{Exponential}(\lambda^2/2) \\ \lambda \sim \text{Gamma}(\alpha, \eta) \\ \text{then } \beta \sim GDP(\tfrac{\eta}{\alpha}, \alpha)\end{gathered}$$

The intuition behind this distribution is such that

- If α grows, the density has lighter tails
- If η grows, the density flattens and variance increases
- If α and η grow at the same rate, variance remains constant, but the tails become lighter. It will approach a Laplace (double exponential) distribution
- The default setup is to have $\alpha = \eta = 1$, which gives a Cauchy-like tail behavior.

Think about the contrasting shapes of the horseshoe prior and the GDP prior. The GDP prior with $\alpha = \zeta = 1$ looks like



This is nice because there is some computational simplicity when using this prior because it places a bit of mass away from 0. Raj showed us a page with the full conditional posterior distributions. They're all in closed form, so implementing a Gibb's sampler should be pretty easy. However, when p is super large, the full conditional for $\boldsymbol{\beta}$ involves inverting a horrendous $p \times p$ matrix. Thus, this is only really effective up to $p \approx 4,000$. Raj notes that when you have a contribution for the posterior where a variable is in the denominator in one exponential term and in the numerator of another exponential term, you're going to find that it follows the inverse Gaussian distribution. He notes on the same page that one could put a prior on α and η such that $\pi(\alpha) = \frac{1}{(1+\alpha)^2}$ and $\pi(\eta) = \frac{1}{(1+\eta)^2}$. However, in this case, you will not have a conjugate full conditional, so you will need to use a M-H step or *griddy Gibbs*.

Alex raised a question about finding the gradient of the distribution and maximizing over the full conditional distribution. This could help in the case of the full conditionals of $\boldsymbol{\beta}$, where you'll be inverting potentially huge matrices. You can basically follow a standard EM algorithm here, finding

$$\boldsymbol{\beta}^{(k+1)} = \arg\max_{\boldsymbol{\beta}} \left\{ -\frac{||\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}||^2}{2\sigma^{(k)2}} - \frac{1}{\sigma^{(k)2}} \sum_{j=1}^{p} \frac{|\beta_j|}{\left(\frac{|\beta_j|^{(k)}}{\sigma^{(k)}} + \eta\right)} \right\}$$

where the superscript k denotes the k^{th} MCMC iteration. This looks suspiciously like Bayesian LASSO.

One can also show that, in a sparse model, $\mathcal{A} = \{\beta_j : \beta_j \neq 0\} \Rightarrow \lim_{n\to\infty} P\left(\mathcal{A}_{GDP} = \mathcal{A}_{true\,model}\right) = 1$.

This will more or less conclude the discussion of shrinkage priors, in which we saw

1. Bayesian LASSO
2. Horseshoe Prior
3. Generalized Double Pareto

Remember, due to computational issues, these methods only really work up to $p = 1000 - 2000$, or at most 5000. So what is this good for? Usually in "Big Data", we really only want good prediction. Parameter estimates are not as important. But say we have $p = 50000$. Is there a Bayesian way to do this? Of course!

2 Compression Matrices

Say we have $n = 400$ and $p = 50000$. We can use something called a **compression matrix**. This will reduce our computational load significantly. We would have something like

$$y = (\Phi x)'\boldsymbol{\beta} + \epsilon$$

where Φ is a $m \times p$ compression matrix. We can set m equal to something like 40 or so. It's sort of like PCA, but not quite. So with large data, we would change things

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} (\Phi x_1)' \\ \vdots \\ (\Phi x_n)' \end{bmatrix} \boldsymbol{\beta} + \epsilon$$

If we say

$$M = \begin{bmatrix} (\Phi x_1)' \\ \vdots \\ (\Phi x_n)' \end{bmatrix}$$

Now we have $\mathbf{y}_{n\times 1} = M_{n\times m}\boldsymbol{\beta}_{m\times 1} + \epsilon_{n\times 1}$. You can choose different Φ, then run each such model and get predictive estimates. If we take the output on all of the models, and take the weighted averages of the prediction, we get pretty good estimates, even with large p. This is a method called **model averaging**. Raj says that if we're interested, we should check out a paper by Guhaniyogi & Dunson (2016).

Compressed Regression

AMS 268

January 29, 2016

For the last week, we have been discussing priors on $\boldsymbol{\beta}$ for the setup

$$y = \mathbf{x}^T\boldsymbol{\beta} + \epsilon,\ \epsilon \sim \mathrm{N}(0, \sigma^2).$$

We were trying to select the prior so we could identify important predictors. In terms of scalability, we were thinking about $p \approx 5,000$.

If we are only interested in prediction and we have predictors $p \approx 50,000$, p is too big for our earlier approaches. We were more interested in parameter estimation in those scenarios. The idea we'll work on today is to bypass parameter estimation and focus solely on prediction for greater scalability.

Goal: Build a predictive model of y on $\mathbf{x}$ where $p \approx 50,000$. Is it possible to create a matrix $\Phi_{m\times p}$ where $m \gg p$ so $\Phi\mathbf{x}$ is an $m \times 1$ vector. We call Φ a **compression matrix**. The idea for Φ came from the compressed sensing literature in computer science. The result was surprising and was published by Johnson and Lindenstrauss in 1984. The basic idea behind that result is that if you have n p-dimensional vectors and a pre-specified level ϵ where $\epsilon \in (0, 1)$ and $m \geq 9\epsilon^{-2}\log(n)$, then there exists a transformation $\underbrace{\Phi\mathbf{x}}_{m\times 1}$ such that, $\forall i, j \in \{1, 2, \ldots, n\}$,

$$||\mathbf{x}_i - \mathbf{x}_j|| - \epsilon \leq ||\Phi\mathbf{x}_i - \Phi\mathbf{x}_j|| \leq ||\mathbf{x}_i - \mathbf{x}_j|| + \epsilon.$$

So the idea is that the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ before the transformation is similar to the distance after the transformation. Choosing an ϵ that is smaller causes m to increase.

Some other researchers found that if you construct Φ so

$$\Phi = \left[\phi_{ij}\right],\ \phi_{ij} \sim \pi$$

then

$$P(||\mathbf{x}_i - \mathbf{x}_j|| - \epsilon < ||\Phi\mathbf{x}_i - \Phi\mathbf{x}_j||_2 < |\mathbf{x}_i - \mathbf{x}_j|| + \epsilon) \geq 1 - e^{-C\epsilon^2 m},$$

where C is some constant depending on ϵ. This result is a bit weaker than Johnson and Lindenstrauss's result. An illustration of what this means can be seen through example. For instance, k-means clustering is a distance-based method, but it cannot be done in high dimensions. In general, this compression method can only be used for methods that are based on distance.

Initially, these ideas were used for

1. k-nearest neighbor clustering
2. Image compression
3. And then people started doing more sophisticated things ...

There is a paper in IEEE (2008) that gives this basic idea:

$$\mathbf{f} = \mathbf{B}\mathbf{w},$$

where $\mathbf{f}$ is a $N \times 1$ vectorized image, $\mathbf{B}$ is a $N \times N$ basis vector, and $\mathbf{w}$ is a $N \times 1$ sparse vector of coefficients (sparse because it is made up of mostly zeros). The idea is that you create

$$\mathbf{y}_{N\times 1} = \Phi_{m\times N}(\mathbf{B}^t\mathbf{f})_{N\times 1},\ m \ll N.$$

After compressing, the goal is to recover $\mathbf{w}$ by finding

$$\arg\min_{\mathbf{w}} \left\{||\mathbf{y} - \mathbf{B}^T\mathbf{f}\mathbf{w}||^2 + \lambda||\mathbf{w}_1||_1\right\} \leftarrow \text{ Raj says this might be wrong, but he'll confirm}$$

The point is that there is some kind of L_1 optimization that needs to be solved. Doing this on the compressed entries lets you find $\mathbf{w}$. They have shown this result in image simulation studies. If the image is sparse in some basis, you can compress the image and then recover the image. In many cases, the wavelet basis (in the frequency dimension) is sparse, so this is often used. Initially, people were using

$$\phi_{ij} \sim \frac{1}{\sqrt{m}}\mathrm{N}(0,1),$$

but now it has been proven that the following works better than the Gaussian:

$$\phi_{ij} = \begin{cases} +1 \text{ with probability } \frac{1}{6} \\ 0 \text{ with probability } \frac{2}{3} \\ -1 \text{ with probability } \frac{1}{6} \end{cases}$$

The idea is that this has to be on the hypersphere. You can also use

$$\phi_{ij} = \begin{cases} \sqrt{3} \text{ with probability } \frac{1}{6} \\ 0 \text{ with probability } \frac{2}{3} \\ -\sqrt{3} \text{ with probability } \frac{1}{6} \end{cases}$$

There is also another choice for ϕ_{ij} on the slides. Note that the machine learning literature never uused this in the regression paradigm. They were only using it in unsupervised cases. Raj extended to the supervised case when the only goal is prediction in his paper *Bayesian Compressed Regression* in JASA (2016). The idea in the paper is

$$\begin{aligned} \mathbf{y} = &\ (\Phi\mathbf{x})^T\mathbf{B} + \epsilon \\ = &\ \mathbf{x}^t\Phi^t\mathbf{B} + \epsilon \end{aligned}$$

so now we're in the standard least squares setting with a conjugate prior (which gives a nice, closed-form posterior). The posteriors come out as

$$\begin{aligned} &\boldsymbol{\beta}|\mathbf{y},\mathbf{y},\Phi \sim \mathrm{N}(\boldsymbol{\mu},\Sigma),\ \boldsymbol{\mu} = \left[\Phi\mathbf{x}^t\mathbf{x}\Phi^T + \Sigma_{\mathbf{B}}^{-1}\right]^{-1}\Phi\mathbf{x}^T\mathbf{y} \\ &\sigma^2|\mathbf{y},\mathbf{x},\Phi \sim \mathrm{IG}(a_1,b_1),\ \Sigma = \left(\tfrac{2b_1}{n}\right)\left[\Phi\mathbf{x}^t\mathbf{x}\Phi^T + \Sigma_{\mathbf{B}}^{-1}\right]^{-1} \\ &a_1 = \tfrac{n}{2},\ b_1 = \tfrac{1}{2}\left[\mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{x}\Phi^T\left[\Phi\mathbf{x}^t\mathbf{x}\Phi^T + \Sigma_{\mathbf{B}}^{-1}\right]^{-1}\Phi\mathbf{x}^T\mathbf{y}\right] \\ &\boldsymbol{\mu}_{pred} = (\Phi\mathbf{x}_{n+1})^T\boldsymbol{\mu} \\ &\sigma^2_{pred} = \tfrac{2b_1}{n}\left[1 + (\Phi\mathbf{x}_{n+1})^T\left[\Phi\mathbf{x}^t\mathbf{x}\Phi^T + \Sigma_{\mathbf{B}}^{-1}\right]^{-1}\Phi\mathbf{x}_{n+1}\right] \end{aligned}$$

The advantage to these closed-form posteriors is that they have no dependency on each other, so no MCMC is required.

Now we face the very important question: How do we choose m (the dimension of Φ)? The answer, for now, is that we rely on **model averaging**. To do this, start with a number of models such that each model corresponds to a dimension of Φ. The posterior probability of model ℓ with data D is

$$P(\text{Model}_\ell|D) = \frac{P(D|\text{Model}_\ell)P(\text{Model}_\ell)}{\sum_{i=1}^{s} P(D|\text{Model}_i)P(\text{Model}_i)},$$

where $P(\text{Model}_i)$ is based on the uniform distribution and s is the total number of models being considered. We'll have

$$\begin{aligned} P(D|\text{Model}_\ell) &= \int \text{N}(\mathbf{y}|\mathbf{x}\Phi\boldsymbol{\beta}, \sigma^2\mathbf{I})\pi(\boldsymbol{\beta}|\sigma^2)\pi(\sigma^2)d\boldsymbol{\beta}d\sigma^2 \\ &= \frac{1}{|\mathbf{X}\Phi^T\Sigma_\beta\Phi\mathbf{x}^T+\mathbf{I}|^{1/2}\left(2^{n/2}\Gamma(\frac{n}{2})/[\mathbf{y}^T(\mathbf{x}\Phi^T\Sigma_\beta\Phi\mathbf{x}^T+\mathbf{I})^{-1}\mathbf{y}]^{1/2}(\sqrt{2\pi})^{n/2}\right)} \end{aligned}$$

As a result of this, the marginal distributions are in closed form. For each model, you try only one Φ and you simulate a new Φ for each model. We do this because Johnson and Lindenstrauss does not hold with probability 1 with our slection of Φ, so for some insurance, we select a new Φ each time.

Next week, we'll talk about

1. g-prior
2. non-parametrics in high-dimensions

g-Prior and high-Dimensional Nonparameterics

AMS 268

February 1, 2016

1 g-Prior

The g-prior was another class of approach that surfaced relatively early due to the ease of computation. If ϕ is a precision parameter. The formulation of the g-prior is

$$\boldsymbol{\beta}|\phi \sim \mathrm{N}(\mathbf{0}, \frac{g}{\phi}(\mathbf{X}'\mathbf{X})^{-1}),\ \pi(\phi) \propto \frac{1}{\phi}.$$

How can we choose g? Can we fix it? Note that $(\mathbf{X}'\mathbf{X})$ is only invertible if $n > p$.

Let the class of models be represented as $\{\mathcal{M}_\gamma : \gamma = (\gamma_1, \ldots, \gamma_p)\}$. The marginal likelihood is given by

$$\pi(\mathbf{y}|\mathcal{M}_\gamma) = \frac{\Gamma((n-1)/2)}{\sqrt{\pi}^{n-1}\sqrt{n}}||\mathbf{y} - \bar{\mathbf{y}}||^{-(n-1)}\frac{(1+g)^{(n-1-p_\gamma)/2}}{[1+g(1-R^2_\gamma)]^{(n-1)/2}},$$

where p_γ is the number of nonzero γ in $\mathcal{M}_\gamma$ and R^2_γ is the R^2 statistic in model $\mathcal{M}_\gamma$. Note that we'll be using this in the case where $n \approx 100$ or 1000 and $p \approx 16$ to 30. What is the Bayes factor here to compare this to the null model?

$$\frac{\pi(\mathbf{y}|\mathcal{M}_\gamma)}{\pi(\mathbf{y}|\mathcal{M}_N)} = \frac{(1+g)^{(n-1-p_\gamma)/2}}{[1+g(1-R^2_\gamma)]^{(n-1)/2}}$$

For $\mathcal{M}_N$, $R^2_N = 0$ and $p_\gamma = 0$. If we set $g \to \infty$while keeping n and p_γ fixed, then the Bayes factor will go to 0. Therefore, the g-prior model will be rejected outright when compared to the null model. This means that an uninformative prior on $\boldsymbol{\beta}$ will help to reject the model even if it is correct. This is known as the **Barlett Paradox**.

If you let $R^2_\gamma \to 1$, then

$$\frac{\pi(\mathbf{y}|\mathcal{M}_\gamma)}{\pi(\mathbf{y}|\mathcal{M}_N)} \to (1+g)^{-p_\gamma/2},$$

which means that the Bayes factor converges to a (finite) constant, even though it should be going to infinity (as it becomes a perfect fit for the data). This is referred to as the **Information Paradox**. You can find both of these paradoxes in (Fi Liang et al. 2008). The idea is that if you choose a fixed g, you're bound for trouble. So, being Bayesians, we'll assigna prior to g. There are basically two priors that are in mainstream use:

1. Zellner-Siow Prior (which is a mixture of g-priors)

$$\pi(g) = \frac{\sqrt{(n/2)}}{\Gamma(1/2)}g^{-(3/2)}e^{-n/(2g)},\ g > 0$$

In this case, the marginal posterior for $\boldsymbol{\beta}$ will be

$$\pi(\boldsymbol{\beta}|\phi) = \int \pi(\boldsymbol{\beta}|g,\phi)\pi(g)dg \propto \frac{\Gamma(p_\gamma/2)}{\pi^{p_\gamma/2}}\left|\frac{\mathbf{X}'\mathbf{X}}{n/\phi}\right|^{1/2} \times (1 + \boldsymbol{\beta}'\frac{\mathbf{X}'\mathbf{X}}{n/\phi}\boldsymbol{\beta})^{-p_\gamma/2}$$

This is a multivariate-t marginal prior.

2. Hyper-g prior

$$\pi(g) = \frac{a-2}{2}(1+g)^{-a/2},\ g > 0$$

This assigns

$$\frac{g}{1+g} \sim \text{Beta}(11, \frac{a}{2} - 1)$$

This also gives a posterior in closed form

$$\pi(g|\mathbf{y}, \mathcal{M}_\gamma) = \frac{(p_\gamma + a - 2)}{2_2F_1\left(\frac{n-1}{2}, 1, \frac{p_\gamma+1}{2}, R^2_\gamma\right)}(1+g)^{(n-1-p_\gamma-a)/2}\left(1 + (1-R^2_\gamma)g\right)^{-(n-1)/2},$$

where

$${}_2F_1(a,b,c,d) = \frac{\Gamma(c)}{\Gamma(b)\Gamma(c-b)}\int_0^1 \frac{t^{b-1}(1-t)^{c-b-1}}{(1-tz)^a}dt$$

Lots of fun. Boy, I sure hope I have to do that sometime in my life.

2 High-dimensional Nonparametrics

Previously, we could have a very pretty true model:

$$y = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p + \epsilon$$

What if we have an ugly true regression equation, like

$$y = \sin(x_1)\beta_1 + e^{x_2}\beta_2 + |x_5 - 0.5|_+\beta_5 + \epsilon?$$

Well, let's rewrite it in standard form:

$$y = \boldsymbol{\mu}_0(bx) + \epsilon$$

where $\boldsymbol{\mu}_0$ is an unknown function, and $\boldsymbol{\mu}_0$ is continuously differentiable up to order s, and $\boldsymbol{\mu}_0^{([\nu])}$ is Lipscitz continuous of order $\nu - \lfloor\nu\rfloor$. This is super-technical. Raj just wanted to convince us that you can't simply throw a complicated Gaussian process like this into a regression setting. A Gaussian process regression can be represented.

Thanos taught us in nonparametrics that we can do this complicated type of regression with the Dirichlet process. The Gaussian process is another way to handle some complex regression like this. We can write

$$y = \boldsymbol{\mu}(\mathbf{x}) + \epsilon$$

with prior on $\boldsymbol{\mu}$

$$\boldsymbol{\mu} \sim \text{GP}(\mathbf{0}, \kappa(\cdot,\cdot,\phi))$$

where

$$\begin{array}{l}\kappa(\mathbf{x}_i, \mathbf{x}_j, \phi) = e^{-||\mathbf{x}_i - \mathbf{x}_j||\phi} \rightarrow \text{ Exponential correlation} \\ \kappa(\mathbf{x}_i, \mathbf{x}_j, \phi) = e^{-||\mathbf{x}_i - \mathbf{x}_j||^2\phi} \rightarrow \text{ Gaussian correlation} \\ \kappa(\mathbf{x}_i, \mathbf{x}_j, \phi) = \text{ Matern class of correlation}\end{array},$$

indexing over the predictors. What this means is that

$$\begin{bmatrix} \mu(\mathbf{x}_1) \\ \vdots \\ \mu(\mathbf{x}_n) \end{bmatrix} \sim \mathrm{N}_n(\mathbf{0}, K)$$

where $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j, \phi)$. Note that this produces an effect very similar to what we did with the Dirichlet process in nonparametrics. Well, what we have essentially written is that

$$\begin{aligned} y_1 &= \boldsymbol{\mu}(\mathbf{x}_1) + \epsilon_1 \\ &\vdots \qquad\qquad\qquad \text{or } \mathbf{y} \sim \mathrm{N}(\boldsymbol{\mu}, \sigma^2\mathbf{I}) \\ y_n &= \boldsymbol{\mu}(\mathbf{x}_n) + \epsilon_n \end{aligned}$$

where

$$\boldsymbol{\mu} \begin{bmatrix} \mu(\mathbf{x}_1) \\ \vdots \\ \mu(\mathbf{x}_n) \end{bmatrix}, \epsilon_i \sim \mathrm{N}(0, \sigma^2).$$

So now let's think about the case in which p is big and the model is sparse. The correlation kernel that we were using before was $\kappa(\mathbf{x}_i, \mathbf{x}_j, \phi) = e^{-||\mathbf{x}_i - \mathbf{x}_j||^2\phi}$. We can change the correlation to be

$$\begin{aligned} \kappa(\mathbf{x}_i, \mathbf{x}_j, \phi, P) = &\quad e^{-(\mathbf{x}_i - \mathbf{x}_j)' P (\mathbf{x}_i - \mathbf{x}_j)\phi} \\ = &\quad e^{-\sum_{k=1}^{p} (x_{ik} - x_{jk})^2 p_{kk} \phi} \end{aligned}$$

where P is a diagonal matrix. If the k^{th} predictor is important, then the posterior of p_{kk} will be distinct from zero. We can give the prior $p_{kk} \sim$ spike-and-slab (this comes from Savitsky et al. 2011). This technique can be combined with compression matrices to work very well with very large number of predictors. This is in the paper from Guhaniyogi et al., 2016, in JMLR. This can't be done for just any data, but $\mathbf{x}$ must lie on a manifold (that is, there must be some kind of correlation between the predictors).

Posterior Consistency

AMS 268

February 3, 2016

Last class we discussed the g-prior and variable selection on a nonparametric Bayes prior,

$$y = \boldsymbol{\mu}(\mathbf{x}) + \epsilon,\ \boldsymbol{\mu} \sim \mathrm{GP}(0, \kappa(\cdot, \cdot, \phi))$$

Where κ is the correlation kernel. This is all covered in the notes from February 1. Today will be the final day for the "small n, big p" paradigm.

Posterior Consistency

When you have a regression problem

$$\begin{gathered} y = \mathbf{X}\boldsymbol{\beta} + \epsilon,\ \epsilon \sim \mathrm{N}(0, \sigma^2) \\ \boldsymbol{\beta} \sim \pi \\ \pi(\boldsymbol{\beta} \in U|\mathbf{y}) = \frac{\int_U f(\mathbf{y}|\boldsymbol{\beta})\pi(\boldsymbol{\beta})d\boldsymbol{\beta}}{\int f(\mathbf{y}|\boldsymbol{\beta})\pi(\boldsymbol{\beta})d\boldsymbol{\beta}} = \frac{\int_U \frac{f(\mathbf{y}|\boldsymbol{\beta})}{f(\mathbf{y}|\boldsymbol{\beta}^0)}\pi(\boldsymbol{\beta})d\boldsymbol{\beta}}{\int \frac{f(\mathbf{y}|\boldsymbol{\beta})}{f(\mathbf{y}|\boldsymbol{\beta}^0)}\pi(\boldsymbol{\beta})d\boldsymbol{\beta}} \end{gathered}$$

Where the true data-generating model is $y = \mathbf{X}\boldsymbol{\beta}^0 + \epsilon$. Let U be a neighborhood of $\boldsymbol{\beta}$ around $\boldsymbol{\beta}^0$. A model is **posterior consistent** if $\pi(\boldsymbol{\beta} \in U|\mathbf{y}) \xrightarrow{n\to\infty} 1$ under $\boldsymbol{\beta}^0$ for any neighborhood U. In other words, as n increases, the posterior distribution should approach the Dirac-δ function with a point mass at $\boldsymbol{\beta}^0$.

Note: depending on the strength of your prior, your posterior will converge at different rates.

Posterior consistency is a sort of analogue of the Law of Large Numbers and the Bayesian CLT.

$$\begin{gathered} x_1, \ldots, x_n \overset{iid}{\sim} \mathrm{N}(\mu, \sigma^2) \\ \text{LLN: } \bar{x} \xrightarrow{a.s.} \mu \text{ as } n \to \infty \\ \text{BCLT: } \sqrt{n}\left(\tfrac{\bar{x}-\mu}{\sigma}\right) \to \mathrm{N}(0, 1) \end{gathered}$$

In the Bayesian paradigm, we also have

$$\pi(\boldsymbol{\beta} \in U|\mathbf{y}) \xrightarrow{n\to\infty} 1 \text{ under } \boldsymbol{\beta}^0$$

Question: How can we choose the neighborhood around $\boldsymbol{\beta}$? We could use

1. Kullback-Leibler neighborhood around $\boldsymbol{\beta}^0$

$$KL(\boldsymbol{\beta}, \boldsymbol{\beta}^0) = \int f(\mathbf{y}|\boldsymbol{\beta}) \log \frac{f(\mathbf{y}|\boldsymbol{\beta}^0)}{f(\mathbf{y}|\boldsymbol{\beta})} dy$$

2. Hellinger neighborhood

$$H(\boldsymbol{\beta}, \boldsymbol{\beta}^0) = \int (\sqrt{f(\mathbf{y}|\boldsymbol{\beta})} - \sqrt{f(\mathbf{y}|\boldsymbol{\beta}^0)})^2 dy$$

3. L-2 Distance

$$||\boldsymbol{\beta} - \boldsymbol{\beta}^0||_2$$

For example, we could choose the neighborhood based on the Kullback-Leibler distance:

$$U = \{\boldsymbol{\beta} : KL(\boldsymbol{\beta}, \boldsymbol{\beta}^2) < \mathcal{E}\}$$

Note that L-2 gives the strongest result. Having convergence in the L-2 neighborhood implies convergence in the other two. The KL and H neighborhoods imply each other, but not L-2. Of course, it's much more difficult to prove convergence in L-2. Note, results supporting consistency using these neighborhoods are forthcoming as time goes on. Raj says to wait a few years, and more results will appear.

If you fix any $\epsilon > 0$, then

$$\pi(U_\epsilon|\mathbf{y}) \xrightarrow{n\to\infty} 1 \text{ under } \boldsymbol{\beta}^0$$

Let's examine this a little more closely. First, say $B_\epsilon = \{\boldsymbol{\beta} : ||\boldsymbol{\beta} - \boldsymbol{\beta}^0||_2 < \epsilon\}$. Let's show that $\pi(B_\epsilon|\mathbf{y}) \xrightarrow{n\to\infty} 1$. First, we'll go back to the posterior:

$$\pi(B_\epsilon|\mathbf{y}) = \frac{\int_{B_\epsilon} \frac{f(\mathbf{y}|\boldsymbol{\beta})}{f(\mathbf{y}|\boldsymbol{\beta}^0)}\pi(\boldsymbol{\beta})d\boldsymbol{\beta}}{\int_{\mathbb{R}^p} \frac{f(\mathbf{y}|\boldsymbol{\beta})}{f(\mathbf{y}|\boldsymbol{\beta}^0)}\pi(\boldsymbol{\beta})d\boldsymbol{\beta}} \xrightarrow{n\to\infty} 1$$

Let $\{\Phi\}_{n=1}^\infty$ be a sequence of test functions for testing $H_0 : \boldsymbol{\beta} = \boldsymbol{\beta}^0$ vs. $H_1 : \boldsymbol{\beta} \in B_\epsilon^c$. We would like to create a sequence of test functions such that

$$E_{\boldsymbol{\beta}^0}(\Phi_n) \leq c_1 \exp(-cn),\ c > 0,\ c_1 > 0$$

and

$$\sup_{\boldsymbol{\beta}\in B_\epsilon^c} E_{\boldsymbol{\beta}}(1 - \Phi_n) \leq b_1 \exp(-bn),\ b > 0,\ b_1 > 0.$$

These are the level and power functions for frequentist tests, respectively. We know that the level of the sequence of tests $\xrightarrow{n\to\infty} 0$ at an exponential rate. and the power function of the sequence of tests $\xrightarrow{n\to\infty} 1$ at an exponential rate. Now we'll use the Borel-Cantelli Lemma (!):

$$\pi(B_\epsilon|\mathbf{y}) = \frac{J_{B_\epsilon}}{J} = \frac{\int_{B_\epsilon} \frac{f(\mathbf{y}|\boldsymbol{\beta})}{f(\mathbf{y}|\boldsymbol{\beta}^0)}\pi(\boldsymbol{\beta})}{\int_{\mathbb{R}^p} \frac{f(\mathbf{y}|\boldsymbol{\beta})}{f(\mathbf{y}|\boldsymbol{\beta}^0)}\pi(\boldsymbol{\beta})} \leq \Phi_n \left(\frac{J_{B_\epsilon}}{J}\right) + \frac{(1-\Phi_n)J_{B_\epsilon}}{J} \leq \Phi_n + (1-\Phi_n)\frac{J_{B_\epsilon}}{J}$$

Note that this is the same as $\pi(B_\epsilon^c|\mathbf{y}) \xrightarrow{n\to\infty} 0$. So Raj will prove that, under $\boldsymbol{\beta}^0$,

- $\Phi_n \xrightarrow{n\to\infty} 0$
- $(1-\Phi_n)J_{B_\epsilon^c}\exp(\frac{bn}{2}) \xrightarrow{n\to\infty} 0$
- $\exp(\frac{bn}{2})J \xrightarrow{n\to\infty} \infty$

We know that $P_{\boldsymbol{\beta}^0}\left(\Phi_n > \exp(-\frac{cn}{2})\right) \leq \frac{E_{\boldsymbol{\beta}^0}(\Phi_n)}{\exp(-\frac{cn}{2})}$ by Chebyshev's inequality. Since we know that $E_{\boldsymbol{\beta}^0}(\Phi_n) \leq c_1\exp(-cn)$, then $P_{\boldsymbol{\beta}^0}\left(\Phi_n > \exp(-\frac{cn}{2})\right) \leq c_1\frac{\exp(-cn)}{\exp(-\frac{cn}{2})} = c_1\exp(-\frac{cn}{2})$. Then we have $\sum_{n=1}^\infty P_{\boldsymbol{\beta}^0}\left(\Phi_n > \exp(-\frac{cn}{2})\right) \leq \sum_{n=1}^\infty c_1\exp(-\frac{cn}{2}) < \infty$. Therefore, by the Borel–Cantelli Lemma, $P_{\boldsymbol{\beta}^0}\left(\Phi_n > \exp(-\frac{cn}{2}) \text{ infinitely often}\right) = 0$. This implies that Φ_n can't exceed $\exp(-\frac{cn}{2})$ as $n \to \infty$ infinitely, which implies that $\Phi_n < \exp(-\frac{cn}{2})$ for all but a finite number of ns. Therefore, $\Phi_n \to 0$.

Say we construct $\Phi_n = I_{||\hat{\boldsymbol{\beta}}-\boldsymbol{\beta}^0||>\epsilon/2}$, then this sequence of tests satisfies $E_{\boldsymbol{\beta}^0} \leq c_1\exp(-cn)$ and $\sup_{\boldsymbol{\beta}\in B_\epsilon^c} E_{\boldsymbol{\beta}}[1-\Phi_n] \leq c_2\exp(-bn)$.

Uh-oh! We are out of time! The fun continues next time.

Posterior Consistency and "Big n, Small p"

AMS 268

February 5, 2016

1 Posterior Consistency

Last time we were talking about posterior consistency. We set

$$B_\epsilon = \{||\beta - \beta^0||_2 < \epsilon\}$$
$$\pi(B_\epsilon|\mathbf{y}) \to 1 \text{ under } \beta^0 \text{ as } n \to \infty$$
$$\iff \pi(B_\epsilon^c|\mathbf{y}) \to 0 \text{ under } \beta^0 \text{ as } n \to \infty$$
$$\pi(B_\epsilon^c|\mathbf{y}) \leq \Phi_n + \frac{(1-\Phi_n)}{J} J_{B_\epsilon^c}$$
$$\text{Where } J_{B_\epsilon^c} = \int_{B_\epsilon^c} \frac{f(\mathbf{y}|\boldsymbol{\beta})}{f(\mathbf{y}|\boldsymbol{\beta}^0)} \pi(\boldsymbol{\beta}) d\boldsymbol{\beta}$$
$$J = \int \frac{f(\mathbf{y}|\boldsymbol{\beta})}{f(\mathbf{y}|\boldsymbol{\beta}^0)} \pi(\boldsymbol{\beta}) d\boldsymbol{\beta}$$

Note that $\{\Phi_n\}_{n\geq 1}$ is an exponentially consistent sequence of test functions for testing $H_0 : \boldsymbol{\beta} = \boldsymbol{\beta}^0$ vs. $H_1 : \boldsymbol{\beta} \in B_\epsilon^c$. We had that $\Phi_n \xrightarrow{n\to\infty} 0$under $\boldsymbol{\beta}^0$. Now we'll show that $(1 - J_{B_\epsilon^c}) \exp(bn/2) \xrightarrow{n\to\infty} 0$. Raj proved this on the board using the Borel-Cantelli Lemma. The end result is that for all but a finite number of ns,

$$\exp(bn/2)(1 - \Phi_n) J_{B_\epsilon^c} \leq \exp(-bn/2)$$

So, $\exp(bn/2)(1 - \Phi_n) J_{B_\epsilon^c} \xrightarrow{n\to\infty} 0$. Next, Raj showed that $\exp(bn/2) J \xrightarrow{n\to\infty} \infty$. These result have been general for all of the shrinkage priors that we have discussed. However, you do need to show that your prior distribution are bounded below by a certain quantity (Given on the board). For certain priors, this is easier than in others.

2 "Big n, Small p"

We were in the "Small n, Big p" setting, but now we're going to change gears. We'll be discussing thefollowing topics over the next three weeks:

1. Sequential MCMC
2. Assumed density filtering/ C-DF
3. Sequential Monte Carlo (SMC)
4. MCMC for big n with stochastic gradient descent
5. Gaussian Processes for big n

Each of these is a big topic, so it we will only give them cursory coverage.

2.1 Sequential MCMC

This is a very important method, and very very new. It's only coming out in the big n setting in the Annals of Statistics in 2016.

Suppose you have a "big" data set or you are dealing with a streaming data set. Requirements (or restrictions) in this setting include:

1. You are not allowed to store the entire data set.
2. The data is being given to you in sets $D_1, D_2, \ldots$. It is not desirable that you store the data at every time point. By this, Raj means that if you are aat time point t, you are not allowed to use the full data set at any given time point t until time point t, i.e. $\{D_1, D_2, \ldots\}$
3. At time point t, you will only use the full D_t, but you can only use summary measures from $D_1, \ldots, D_{t-1}$

At every time point, you will update and propagate those summary measures. Note that the parameter space needs to be fixed for all of the time points.

2.1.1 Algorithm

1. Assume that you have decided to fit a model to the data and the model involves a parameter $\boldsymbol{\theta}$.
2. Let $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_k)$ and $\theta_i|\boldsymbol{\theta}_{-i}$ has a recognizable form that you can easily sample from. At time t, the conditional posterior would be $\theta_i|\boldsymbol{\theta}_{-i}, D_1, D_2, \ldots, D_t$, but at time $t+1$, the conditional posterior would be $\theta_i|\boldsymbol{\theta}_{-i}, D_, \ldots, D_{t+1}$, so the conditional posterior keeps changing.
3. Let's assume that $\theta_i|\boldsymbol{\theta}_{-i}, D_1, D_2, \ldots, D_t$ is dependent on $D_1, D_2, \ldots, D_t$ only through a summary statistic. Raj showed this in the case of a normal-normal update.
4. At time $t+1$, obtain D_{t+1}
5. Update the sufficient statistics

   $$C_{1,t+1} = g(C_{1,t}, D_{t+1}), \ldots, C_{m,t+1} = g(C_{m,t}, D_{t+1})$$

   from some function g dependent on the model. Remember, you may have more than one sufficient statistic.
6. Draw n_{t+1} MCMC samples from the conditional distributions
7. Set $t+1 \to t+2$ and carry out the above steps again!

Methods for "Big n, Small p"

AMS 268

February 8, 2016

1 Sequential MCMC

We began talking about the algorithm for sequential MCMC

1.1 Algorithm

1. Assume that you have decided to fit a model to the data and the model involves a parameter $\boldsymbol{\theta}$.
2. Let $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_k)$ and $\theta_i|\boldsymbol{\theta}_{-i}$ has a recognizable form that you can easily sample from. At time t, the conditional posterior would be $\theta_i|\boldsymbol{\theta}_{-i}, D_1, D_2, \ldots, D_t$, but at time $t+1$, the conditional posterior would be $\theta_i|\boldsymbol{\theta}_{-i}, D_, \ldots, D_{t+1}$, so the conditional posterior keeps changing.
3. Let's assume that $\theta_i|\boldsymbol{\theta}_{-i}, D_1, D_2, \ldots, D_t$ is dependent on $D_1, D_2, \ldots, D_t$ only through a summary statistic. Raj showed this in the case of a normal-normal update.
4. At time $t+1$, obtain D_{t+1}
5. Update the sufficient statistics

   $$C_{1,t+1} = g(C_{1,t}, D_{t+1}), \ldots, C_{m,t+1} = g(C_{m,t}, D_{t+1})$$

   from some function g dependent on the model. Remember, you may have more than one sufficient statistic.
6. Draw n_{t+1} MCMC samples from the conditional distributions
7. Set $t+1 \to t+2$ and carry out the above steps again!

So we need to "combine" the summary statistics from all past time steps and the current time step. The statistic used for this will determine the method used to combine the results.

1.1.1 Markov Chain Convergence Consistency

1. If your posterior π_t changes slowly over time as $t \to \infty$
2. The transition kernel is "ergodic", then $||T_t(\cdot, \cdot) - \pi_t|| < e\epsilon$ for some $\epsilon \in (0, 1)$.
3. Then the Sequential MCMC samples will correspond to samples from π_t at large t

   $$T_1^{n_1} T_2^{n_2} \cdots T_t^{n_t} - \pi(t)||_{TV} \to 0$$

   where T_t is the transition kernel at time t.

Sequential MCMC, as it stands today, doesn't have much room for growth. We could try it within assumed-density filtering, but nobody has tried it yet (but Raj says he will).

2 Sequential Monte Carlo

This is based on the idea of importance sampling. The main idea is to calculate $E[h(\boldsymbol{\theta})]$ through the evaluation of the integral $\frac{\int h(\boldsymbol{\theta})\pi(\boldsymbol{\theta})}{\pi(\boldsymbol{\theta})}$. Now say it is difficult to sample from π, but easy to sample from a distribution g. We could then write

$$\frac{\int h(\boldsymbol{\theta})\pi(\boldsymbol{\theta})}{\int \pi(\boldsymbol{\theta})} = \frac{\int h(\boldsymbol{\theta})\frac{\pi(\boldsymbol{\theta})}{g(\boldsymbol{\theta})}g(\boldsymbol{\theta})}{\int \frac{\pi(\boldsymbol{\theta})}{g(\boldsymbol{\theta})}g(\boldsymbol{\theta})}$$

If $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_p$ are samples from g, this integral can be approximated as

$$\frac{\sum_{j=1}^{p} h(\boldsymbol{\theta}_j)\frac{\pi(\boldsymbol{\theta}_j)}{g(\boldsymbol{\theta}_j)}}{\sum_{j=1}^{p} \frac{\pi(\boldsymbol{\theta}_j)}{g(\boldsymbol{\theta}_j)}}$$

However, when g is not a good approximation of π, then you are screwed.

2.1 Algorithm

1. Draw samples from π_0. π_0 is the prior distribution for $\boldsymbol{\theta}$.
2. You now have D_1
3. You have to draw samples from π_1
4. Use importance sampling to draw samples from π_1
5. Get D_2
6. Use importance sampling with $g = \pi_1$ to draw samples from π_2
7. And so on ...

2.2 Issues

This depends on the order of the data. It will result in something called **particle degeneration**. This means that one particle will become disproportionally "important", and we'll end up resampling from this too often.

Sequential Monte Carlo and Assumed Density Filtering

AMS 268

February 10, 2016

We'll be talking about Sequential Monte Carlo in fixed parameter settings. The first big breakthrough came from Chopin in 2002. Raj notes that this is a European-dominated area of research.

1 Sequential Monte Carlo - ISIS Algorithm

1. Lets say that with every batch of data, you have p samples, i.e. the first batch is $(y_1, \ldots, y_p)$, the second batch is $(y_{p+1}, \ldots, y_{2p})$, and so on
2. Draw s samples from π_{init}, which is either the prior distribution, or the posterior distribution from the first batch of data.
3. Let these samples be $\theta_1, \ldots, \theta_s$
4. Propagate these samples
5. Let at time t, you have samples $\theta_1^t, \ldots, \theta_s^t$
6. Observe the $(t+1)^{th}$ batch of data
7. Calculate $w_j \propto \frac{\pi(\boldsymbol{\theta}_j^t | D_1, \ldots, D_{t+1})}{\pi(\boldsymbol{\theta}_j^t | D_1, \ldots, D_t)} \propto f(D_{t+1} | \boldsymbol{\theta}_j^t)$ where D_t is the data batch at time t.
8. Now look at the discrete distribution $\theta_1^t, \ldots, \theta_s^t$ with probabilities $w_1, \ldots, w_s$
9. We will resample (with replacement) s samples from this distribution to get the important particles $(h_1, \ldots, h_s)$. This resampling step is going to get rid of unimportant particles, but it reduces the number of particles significantly.
10. Now we will use a step known as the "particle rejuvenation" step.
    $\boldsymbol{\theta}_j^{t+1} \sim k(\cdot, h_j)$ where k is a transition kernel whose stationary distribution is π_{t+1}.
11. The proposal distribution for a Metropolis-Hastings step is

$$\mathrm{N}(\boldsymbol{\mu}, \Sigma),\ \boldsymbol{\mu} = \frac{\sum_{j=1}^s \boldsymbol{\theta}_j^t}{s},\ \Sigma = \frac{1}{s}\sum_{j=1}^s (\boldsymbol{\theta}_j^t - \boldsymbol{\mu})(\boldsymbol{\theta}_j^t - \boldsymbol{\mu})'$$

    This is used because it have a very high probability of moving in each iteration.

Question: What happens when $\boldsymbol{\theta}_j$ has dimension $k \times 1$ when k is big? That is, what happens if the parameter space is big?

The particles will not change very much over time, so the entire system becomes degenerate very quickly. This hasn't been proven, but it has been observed. To avoid this issue, one can start with a large number of particles, i.e. s has to be large.

In the particle rejuvenation step, we are carrying out one-step MCMC. This means we have to calculate the likelihood of $D_1, \ldots, D_{t+1}$. Therefore, we need to store all batches of data up to time $t+1$.

Another issue: In this algorithm, it matter how you order your data. This doesn't matter so much when you have streaming data, but in the case where you are using it for "big" data, this is a big criticism. What people have tried to do is randomly partition the data multiple times and run SMC to show the potential errors that may occur.

Third issue: How should p be chosen? If the model is complex, then at each time point, feed the model with more data.

2 Assumed Density Filtering

Setting: The posterior is complex. You have to approximate the posterior at every time point (we're talking streaming data here) with a density from a suitable class of distributions. In contrast, we use the Normal distribution with the Laplace Approximation, but in this case, the proposal does not need to be normal.

Let $D_1, \ldots, D_t, \ldots$ be the batches of data. Assume a class of distributions used to approximate the posterior. Let's call this class $Q = \{q_\eta : \eta \in E\}$. For example, $Q = \{\mathrm{N}(\boldsymbol{\mu}, \Sigma) : \boldsymbol{\mu} \in \mathbb{R}^p, \Sigma \in \mathbb{R}^{\frac{p(p+1)}{2}}\}$

2.1 Algorithm

1. Let's start with the first batch of data D_1. $\pi(\boldsymbol{\theta}|D_1)$ is not in a closed form that you can sample from.
2. Take the Kullback-Liebler divergence such that $\arg\min_\eta KL(\pi(\boldsymbol{\theta}|D_1)||q_\eta(\boldsymbol{\theta})) = \eta^*$ which gives us the distribution $q_{\eta^*}(\boldsymbol{\theta})$
3. Use this as an approximate posterior from time 1, and we'll use it as the prior of $\boldsymbol{\theta}$ at time $t = 2$
4. Calculate a pseudoposterior
$$\pi^{ps}(\boldsymbol{\theta}|D_1, D_2) = \frac{q_{\eta^*}(\boldsymbol{\theta}) f(D_2|\boldsymbol{\theta})}{\int q_{\eta^*}(\boldsymbol{\theta}) f(D_2|\boldsymbol{\theta}) d\boldsymbol{\theta}}$$
5. Once again, find $\arg\min_\eta KL(\pi^{ps}(\boldsymbol{\theta}|D_1, D_2)||q_\eta(\boldsymbol{\theta})) = \eta^{**}$
6. Now use $q_{\eta^{**}}(\boldsymbol{\theta})$ as the approximate posterior after time $t = 2$.
7. And so on ...

Assumed Density Filtering

AMS 268

February 12, 2016

1 Assumed Density Filtering

Last time, we began talking about Assumed Density Filtering (ADF). Let's clarify the setting a little more.

$$y_i|\mathbf{U} \sim P(y_i|\mathbf{U}),\ i = 1, \ldots, n$$
$$\mathbf{U} \sim P^{(0)}(\mathbf{U})$$

We have two quantities of interest:

1. The posterior over the latent variable $P(\mathbf{U}|D)$
2. The likelihood fo the observation

We require that the approximated posterior is of the same class as $P^{(0)}(\mathbf{U})$. So, say that in each time step we have

$$t_0(\mathbf{U}) = P^{(0)}(\mathbf{U})$$
$$t_i(\mathbf{U}) = P(y_i|\mathbf{U})$$
$$\text{and therefore } \ldots$$
$$P(D, \mathbf{U}) = \prod_{i=0}^{n} t_i(\mathbf{U})$$

When $P(y_i|\mathbf{U})$ is the density of a mixture of Gaussians, one generally sets the approximated class to Gaussian. So start with a class of distributions that we want to approximate the posterior with. Let q^{-i} be the old approxiamtion of the posterior at time i, which is found before y_i has been observed. Then, at each stage, compute

$$\hat{p}(\mathbf{U}) = \frac{t_i(\mathbf{U})q^{-i}(\mathbf{U})}{\int t_i(\mathbf{Z})q^{-i}(\mathbf{Z})d\mathbf{Z}}$$

The idea is to minimize the KL-divergence $KL(\hat{p}(\mathbf{U})||q(\mathbf{U})$ subject to the condition that q is in the prespecified approximating family. For example, when the approximating class is Gaussian, and $P(\mathbf{y}_i|\mathbf{U})$ belongs to the exponential family, then the best approximating class is the Gaussian distribution. We can find the specific Gaussian distribution by matching moments, i.e.

$$E_{\hat{p}}(\mathbf{U}) = E_q(\mathbf{U})$$
$$E_{\hat{p}}(\mathbf{U}\mathbf{U}') = E_q(\mathbf{U}\mathbf{U}')$$

1.1 Alternative view

ADF can be described as treating each obsercation term t_i exactly and then approximating the posterior including t_i. Another way to think of it is as first approximating each t_i by $\tilde{t}_i$ and then compute the exact posterior using $\tilde{t}_i$. This interpretation is always possible as we think of $\tilde{t}_i = \frac{q}{q^{-i}}\mathbf{Z}_i$ where $\mathbf{Z}_i$ is constant. Importantly, if the approximate posterior is in the exponential family, the term will also be in the same family.

ADF is viewed as the approximated likelihood with the exact likelihood. Let's look at it as an exact posterior with approximated likelihood. This was proposed by Minka at Microsoft Research in 2009.

So $\tilde{t}_i(\mathbf{U})$ is the approximated likelihood contribution from time i. If the approximating class is in the exponential family, then both $q(\mathbf{U})$ and $q^{-i}(\mathbf{U})$ will be in the same family, therefore, $\tilde{t}_i(\mathbf{U})$ will also be in the same family.

1.2 Issues

- ADF makes one pass through the data, so $\tilde{t}_i$ is only approximated one time, so the earlier approximations may be poor, which you cannot go back and fix.
- Depends on the order in which the data is processed, in principal. It would be better to update earlier approximations at a later time, i.e. have $\tilde{t}_i$ get updated multiple times. EP (Expectation Propagation) effectively extends ADF by allowing multiple passes through the data.

2 Expectation Propagation

This is very similar to the idea behind ADF, but now the algorithm is:

1. Initialize $\tilde{t}_i$
2. Compute $q(\mathbf{U}) = \frac{\prod_i \tilde{t}_i(\mathbf{U})}{\int \prod_i \tilde{t}_i(\mathbf{Z})d\mathbf{Z}}$
3. Until all $\tilde{t}_i$ converges:
   (a) Choose a $\tilde{t}_i$ to refine
   (b) Remove $\tilde{t}_i$ from the posterior to get an old posterior q^{-i} by dividing and normalizing
   $$q^{-i}(\mathbf{U}) \propto \frac{q(\mathbf{U})}{\tilde{t}_i(\mathbf{U})}$$
   (c) Combine $q^{-i}(\mathbf{U})$ and $t_i(\mathbf{U})$ as
   $$\hat{p}(\mathbf{U}) = \frac{t_i(\mathbf{U})q^{-i}(\mathbf{U})}{\int t_i(\mathbf{U})q^{-i}(\mathbf{U})d\mathbf{U}}$$
   and minimize the KL divergence to get a new posterior $q(\mathbf{U})$ with normalizer Z_i.
   (d) Update $\tilde{t}_i = Z_i q^{-i}/q$
4. Use the normalizing constant of $q(\mathbf{U})$ as an approximation to $P(D)$, $P(D) \approx \int \prod \tilde{t}_i(\mathbf{z})d\mathbf{z}$.

So before, in ADF, at time $i-1$, the approximate posterior $q^{-i}(\mathbf{U})$ is combined with $\tilde{t}_i$ to find

$$q(\mathbf{U}) \propto \frac{t_i(\mathbf{U})q^{-i}(\mathbf{U})}{\int t_i(\mathbf{Z})q^{-i}(\mathbf{Z})d\mathbf{Z}}$$

So now, basically, q^{-i} means the the approximation based on **all** other time points, instead of all earlier time points.

ADF and Expectation Propagation

AMS 268

February 17, 2016

1 Expectation Propagation

1.1 Example

Let's say that our likelihood is a mixture of normals:

$$p(y|\mathbf{U}) = (1-w)\mathrm{N}(\mathbf{y}|\mathbf{U},\mathbf{I}) + w\mathrm{N}(\mathbf{y}|\mathbf{0},10\mathbf{I})$$

where w is known and $\mathbf{U}$ is $d \times 1$. We'll begin by saying that

$$p(\mathbf{U}) = \mathrm{N}(\mathbf{U}|\mathbf{0},100\mathbf{I}_d)$$

and we have data

$$D = \{\mathbf{y}_1,\ldots,\mathbf{y}_n\}$$

Goal: Find the posterior $p(\mathbf{U}|\mathbf{y}_1,\ldots,\mathbf{y}_n)$. We want to find an approximating posterior using ADF. We'll do this using the following class of distributions:

$$\{q : q(\mathbf{U}) = \mathrm{N}\left(\mathbf{U}|\mathbf{m}_U, v_n\mathbf{I}_d\right)\}$$

where $\mathbf{m}_U$ is $d \times 1$ and $v_n > 0$. So we can establish that, at time i,

$$q^{-i}(\mathbf{U}) = \mathrm{N}(\mathbf{U}|\mathbf{m}_U^{-i}, v_U^{-i}\mathbf{I}_d)$$

So we have that

$$\begin{aligned} Z_i &= \int p(\mathbf{y}_i|\mathbf{U})q^{-i}(\mathbf{U})d\mathbf{U} \\ &= \int \left[(1-w)\mathrm{N}(\mathbf{y}_i|\mathbf{U},\mathbf{I}) + w\mathrm{N}(\mathbf{y}_i|\mathbf{0},10\mathbf{I})\right]\mathrm{N}(\mathbf{U}|\mathbf{m}_U^{-i}, v_U^{-i}\mathbf{I}_d)d\mathbf{U} \\ &= (1-w)\mathrm{N}(\mathbf{y}_i|\mathbf{m}_U^{-i},(v_U^{-i}+1)\mathbf{I}) + w\mathrm{N}(\mathbf{y}_i|\mathbf{0},10\mathbf{I}) \end{aligned}$$

So the idea is to find

$$\begin{aligned} \hat{p}(\mathbf{U}) &= \frac{p(\mathbf{y}_i|\mathbf{U})q^{-i}(\mathbf{U})}{\int p(\mathbf{y}_i|\mathbf{U})q^{-i}(\mathbf{U})d\mathbf{U}} \\ &= \frac{p(\mathbf{y}_i|\mathbf{U})q^{-i}(\mathbf{U})}{Z_i} \\ &= \frac{\left[(1-w)\mathrm{N}(\mathbf{y}_i|\mathbf{U},\mathbf{I}) + w\mathrm{N}(\mathbf{y}_i|\mathbf{0},10\mathbf{I})\right]\mathrm{N}(\mathbf{U}|\mathbf{m}_U^{-i}, v_U^{-i}\mathbf{I}_d)}{Z_i} \end{aligned}$$

So $\hat{p}(\mathbf{U})$ is the approximate density with a spherical Normal distribution $q(\mathbf{U})$. This approximation is with respect to the KL divergence

$$\Rightarrow E_{\hat{p}}[\mathbf{U}] = E_q[\mathbf{U}]$$
$$E_{\hat{p}}[\mathbf{U}\mathbf{U}'] = E_q[\mathbf{U}\mathbf{U}']$$

And we can solve these two equations to obtain the updated mean $\mathbf{m_U}$ and covariance $v_\mathbf{U}\mathbf{I}_d$ in the Normal. In fact, if you solve these two equations, you will find

$$\begin{aligned}
\mathbf{m}_U &= \mathbf{m}_U^{-i} + v_U^{-i} r_i \frac{(\mathbf{y}_i - \mathbf{m}_U^{-i})}{v_U^{-i}} \\
r_i &= 1\frac{w}{Z_i}\mathrm{N}(\mathbf{y}_i|\mathbf{0}, 10\mathbf{I}_d) \\
v_U &= v_U^{-i} - \frac{r_i (v_U^{-i})^2}{v_U^{-i}+1} + r_i(1-r_i)(v_U^{-i})^2 \frac{||\mathbf{y}_i - \mathbf{m}_U^{-i}||^2}{d(v_U^{-i}+1)^2}
\end{aligned}$$

So we'll initialize $\mathbf{m}_U = \mathbf{0}$ and $v_U = 100$, and say the normalizing constant $S = 1$, that is, at the prior value.

1. For each data block $\mathbf{y}_i$, update $\mathbf{m}_U$ and v_U, $S = S^{-i}Z_i$, where S^{-i} is the normalizing constant at the time just before i.
   When we have

   $$\tilde{t}_i(\mathbf{U}) \propto \frac{q(\mathbf{U})}{q^{-i}(\mathbf{U})}$$

   and both q and q^{-i} are Gaussian, then $\tilde{t}_i$ is also Gaussian and, more specifically,

   $$S_i \exp\left[-\frac{1}{2v_i}(\mathbf{U}-\mathbf{m}_i)'(\mathbf{U}-\mathbf{m}_i)\right]$$

   In the initial case, $v_0 = 100$, $\mathbf{m}_0 = \mathbf{0}$, $S_0 = (2\pi v_0)^{-d/2}$.
2. $\mathbf{m}_U = \mathbf{m}_0$, $v_U = v_0$
3. $\tilde{t}_i(\mathbf{U}) = 1$, $i \geq 1$, $\mathbf{m}_i = \mathbf{0}$, $v_i = \infty$, $S_i = 1$. Until $(\mathbf{m}_i, v_i, S_i)$ converge:

   a) For $i = 1, \ldots, n$, remove $\tilde{t}_i(\mathbf{U})$ and then $(\mathbf{m}_U, v_u) \to (\mathbf{m}_U^{-i}, v_u^{-i})$ by:

   $$\begin{aligned}
   (v_U^{-i})^{-1} &= v_U^{-1} - v_i^{-1} \\
   \mathbf{m}_U^{-i} &= \mathbf{m}_U + \frac{v_u^{-i}}{v_i}(\mathbf{m}_u - \mathbf{m}_i) \\
   q^{-i}(\mathbf{U}) &= \mathrm{N}(\mathbf{m}_U^{-i}, v_U^{-i}\mathbf{I}_d) \\
   p(\mathbf{y}_i|\mathbf{U}) &\to q(\mathbf{U})
   \end{aligned}$$

Statistical Modeling for Big n

AMS 268

February 19, 2016

We will mainly be talking about nonparametric Gaussian Process-based models for big n. Consider the scenario:

$$y = \mu_0(\mathbf{x}) + \epsilon \qquad\qquad \epsilon \sim \mathrm{N}(0, \sigma^2)$$

where μ_0 is some unknown function of the covariates. For example, $\mu_0(\mathbf{x}) = 3x_1 + e^{x_2} + 9\log x_3$. So what we'll be working with is coming up with a model

$$y = \mu(\mathbf{x}) + \epsilon \qquad\qquad \epsilon \sim \mathrm{N}(0, \sigma^2)$$

where μ is an unknown function and the idea is to estimate the unknown function and σ^2. What do we need?

First, we'll need a prior distribution on σ^2, which is parametric. We'll also need a prior distribution on μ, which is a **stochastic process**. We can use the Dirichlet Process, but we'll be using the Gaussian Process in this course because it is computationally easier. Let's begin with the development of the Gaussian Process prior:

$$\mu \sim \mathrm{GP}(m(\cdot), \kappa(\cdot, \cdot))$$

$m(\cdot)$ is a known **mean function** where you will center the GP. In many applications, $m(\cdot) = 0$. $\kappa(\cdot, \cdot)$ is the **covariance kernel**, which is a function that specifies covariance between $\mu(\mathbf{x})$ and $\mu(\mathbf{z})$, which could be thought of as $\mathrm{Cov}(\mu(\mathbf{x}), \mu(\mathbf{z})) = \kappa(\mathbf{x}, \mathbf{z})$. The covariance kernel determines the "smoothness" of the prior realization. There are two ways to measure smoothness: Holder α-continuity and Sobolev continuity. Sometimes, the covariance kernel will only depend on the difference between them. Sometimes, it is just based on the difference between them (note: not the same thing).

$$\kappa(\mathbf{x}, \mathbf{z}) = K(||\mathbf{x} - \mathbf{z}||)$$
$$\text{or } K(\mathbf{x} - \mathbf{z})$$

These are known as stationary covariance kernels. If $\kappa(\mathbf{x}, \mathbf{z})$ does not depend on $\mathbf{x} - \mathbf{z}$, then it is referred to as a *nonstationary kernel*. A few other kernels are

$$\kappa(\mathbf{x}, \mathbf{z}, \phi, \tau^2) = \tau^2 \exp\{-\phi||\mathbf{x} - \mathbf{z}||\} \qquad \leftarrow \text{ exponential kernel}$$
$$\kappa(\mathbf{x}, \mathbf{z}, \phi, \tau^2) = \tau^2 \exp\{-\phi||\mathbf{x} - \mathbf{z}||^2\} \qquad \leftarrow \text{ Squared exponential/Gaussian kernel}$$
$$\kappa(\mathbf{x}, \mathbf{z}, \nu, \phi, \tau^2) = \frac{\tau^2(\phi||\mathbf{x} - \mathbf{z}||)^2}{\Gamma(\nu)2^{\nu-1}}\kappa_\nu(\phi||\mathbf{x} - \mathbf{z}||) \qquad \leftarrow \text{ Matérn kernel}$$

Where κ_ν is the modified Bessel function of the second kind. Stein, in his book from 1999, defined the following:

$\nu =$ smoothness parameter
$\phi =$ range parameter
$\tau^2 =$ spatial variability

Let's examine some special cases,

take $\nu = \frac{1}{2}$ in the Matérn kernel, then $\kappa(\mathbf{x}, \mathbf{z}, \frac{1}{2}, \tau^2, \phi) = \tau^2 \exp\{-\phi||\mathbf{x} - \mathbf{z}||\}$, the exponential kernel.
take $\nu = \infty$ in the Matérn kernel, then $\kappa(\mathbf{x}, \mathbf{z}, \infty, \tau^2, \phi) = \tau^2 \exp\{-\phi||\mathbf{x} - \mathbf{z}||^2\}$, the Gaussian kernel

For any ν, the realizations of Gaussian process with Matérn kernel is $\lfloor \nu \rfloor$-times continuously differentiable. If $\lfloor \nu \rfloor = 0$, then the realizations from the exponential kernel are not even once differentiable, so you will get "rough" realizations. By contrast, the Gaussian kernel gives infinitely differentiable realizations, which would be "smoother". (Small note: The derivative of Gaussian Processes are also Gaussian Processes) So what role does ϕ play in an exponential kernel? Think of it as a "fade" parameter. Since $\phi > 0$, a small ϕ means that only $\mathbf{x}$ and $\mathbf{z}$ that are close together have a covariance.

Let's get back to the Gaussian Process specification.

$$\mu(\cdot) \sim \text{GP}(m(\cdot), \kappa(\cdot, \cdot, \boldsymbol{\theta})) \qquad \boldsymbol{\theta} = (\tau^2, \phi, \nu)$$

Hao Zhang, in 2004, proved that τ^2, ϕ, ν cannot be consistently estimated together. $\tau^2\phi^{2\nu}$ can be consistently estimated, however. Basically, in any practical analysis, fix ν. You have to estimate ϕ and τ^2. Carry out empirical estimation of the range of ϕ, then place a uniform prior on the range of ϕ that you found empirically ($\phi \sim \text{U}(a, b)$ using empirical variogram). We'll learn what an empirical variogram is in some future course.

We have the model

$$y = \mu(\mathbf{x}) + \epsilon \qquad \epsilon \sim \text{N}(0, \sigma^2)$$

$$y_1, \ldots, y_n$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} \mu(\mathbf{x}_1) \\ \vdots \\ \mu(\mathbf{x}_n) \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{bmatrix} \qquad \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{bmatrix} \sim \text{N}(0, \sigma^2\mathbf{I})$$

So what we say is that $\mu(\mathbf{X}) \sim \text{MVN}(\mathbf{0}, \kappa)$, and we can rewrite the distribution of the likelihood is $\text{MVN}(\mathbf{y}|\mathbf{0}, \sigma^2\mathbf{I} + \kappa)$. If we put inverse gamma prior distributions on σ^2 and τ^2, and a uniform on ϕ. This means that at every MCMC iteration, you would have to invert an $n \times n$ matrix, which takes a long time if n is large, as it is in our case. This is known as a "big n" problem. There are a few approaches to solve this problem:

1. Low rank models
2. Class of sparsity-based models

One way to deal with this with low-rank models is the **Sherman-Woodbury-Morrison Matrix Identity**.

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U\left(C^{-1} + VA^{-1}U\right)^{-1}VA^{-1}$$

Gaussian Processes

AMS 268

February 22, 2016

We have the model

$$y = \mu(\mathbf{x}) + \epsilon \qquad \epsilon \sim \mathrm{N}(0, \sigma^2)$$

$$y_1, \ldots, y_n$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} \mu(\mathbf{x}_1) \\ \vdots \\ \mu(\mathbf{x}_n) \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{bmatrix} \qquad \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{bmatrix} \sim \mathrm{N}(0, \sigma^2\mathbf{I})$$

and $\mu(\mathbf{X}) \sim \mathrm{MVN}(\mathbf{0}, \kappa)$. We rewrote the likelihood as $\mathrm{MVN}(\mathbf{y}|\mathbf{0}, \sigma^2\mathbf{I} + \kappa)$, which results in a terrible matrix-inversion bottleneck when n is large. We discussed two methods to deal with this problem:

1. Low rank methods
2. Class of sparsity-based models

1 Kernel Convolution (Higdon 2001)

This is one of the low rank methods.

$$\mu(\mathbf{x}) = \int K(\mathbf{x} - \mathbf{u}) z(\mathbf{u}) d\mathbf{u}$$

If $\mu(\mathbf{x})$ is a Gaussian, it can be written in the above way where $K(\cdot)$ is some function and $z(\mathbf{u})$ is a white noise process. Now let's start by choosing, in some way, some p-dimensional vectors $\mathbf{u}_1^*, \ldots, \mathbf{u}_m^*$. Now we can approximate

$$\mu(\mathbf{x}) \approx \sum_{\ell=1}^{m} K(\mathbf{x} - \mathbf{u}_\ell^*) z(\mathbf{u}_\ell^*)$$

Now we can fit the approximation

$$y_i \approx \sum_{\ell=1}^{m} K(\mathbf{x} - \mathbf{u}_\ell^*) z(\mathbf{u}_\ell^*) + \epsilon_i$$

This is much easier to fit, and we don't have the same matrix inversion issue when we have large n. The matrix form of the model is

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} K(\mathbf{x}_1 - \mathbf{u}_1^*) & \cdots & K(\mathbf{x}_1 - \mathbf{u}_m^*) \\ \vdots & & \vdots \\ K(\mathbf{x}_n - \mathbf{u}_1^*) & \cdots & K(\mathbf{x}_n - \mathbf{u}_m^*) \end{bmatrix} \begin{bmatrix} z(\mathbf{u}_1^*) \\ \vdots \\ z(\mathbf{u}_m^*) \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

Which is $\mathbf{y} = M\mathbf{z} + \boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim \mathrm{N}(\mathbf{0}, \sigma^2\mathbf{I})$, $\mathbf{z} \sim \mathrm{N}(\mathbf{0}, \sigma_z^2\mathbf{I})$. Integrating out the $\mathbf{z}$, we find

$$\pi(-|\mathbf{y}) \propto \mathrm{N}(\mathbf{y}|\mathbf{0}, M\Sigma M' + \sigma^2\mathbf{I}) \times \pi(-)$$

$$\Sigma = \sigma_z^2\mathbf{I}$$

Now the new likelihood has to simulated at each MCMC iteration, meaning a matrix inversion is necessary

$$\begin{aligned}\Rightarrow (M\Sigma M' + \sigma^2\mathbf{I})^{-1} &= \frac{1}{\sigma^2}\mathbf{I} - \frac{1}{\sigma^2}M\left(\frac{1}{\sigma_z^2}\mathbf{I} + M'\frac{1}{\sigma^2}M\right)^{-1}M'\frac{1}{\sigma^2} \\ &= \frac{1}{\sigma^2}\mathbf{I} - \frac{M}{\sigma^2}\left(\frac{1}{\sigma_z^2}\mathbf{I} + \frac{M'M}{\sigma^2}\right)^{-1}M'\frac{1}{\sigma^2}\end{aligned}$$

so matrix that needs to be inverted is now $m \times m$, as opposed to $n \times n$. This leads to a trade-off

- if m is small $\Rightarrow$ poor approximation to the GP
- if m is large $\Rightarrow$ computational issues as the matrix to be inverted gets larger and larger.

Note that the choice of K is arbitrary. How does one choose $\mathbf{u}^*$? Some say to do it on a grid of points, where vertices are referred to as *knots*. However, Raj says we can also choose values for $\mathbf{u}^*$ randomly using the uniform distribution. The problem in these scenarios is that we do not know which K should be used. Some people use wavelet basis functions. Raj says the breakthrough paper in this field came out in 2008 that eliminated this arbitrary choosing of basis functions. That paper is *Gaussian Predictive Process Model (JRSS-B, 2008)* by Banerjee, Gelfand, Finley, & Sang.

2 Gaussian Predictive Process Model

We still have to choose knot points in the space. Let's call them $s_1^*, \ldots, s_m^*$. Now call $w(s)$ the GP that you would like to fit to your data. So our intention is to fit this model:

$$y = \mu(s) + \epsilon,\ \mu \sim \mathrm{GP}(0, \kappa(\cdot,\cdot))$$

What we'll do to approximate this difficult model is we'll fit $y = \tilde{\mu}(s) + \epsilon$ where $\tilde{\mu}(s) = \mathrm{E}\left[\mu(s)|\mu(s_1^*, \ldots, s_m^*)\right]$. Let's examine $\tilde{\mu}(s)$ more closely. Since μ is a Gaussian process,

$$\begin{bmatrix}\mu(s) \\ \mu(s_1^*) \\ \vdots \\ \mu(s_m^*)\end{bmatrix} \sim \mathrm{N}\left(\mathbf{0},\ \begin{bmatrix}\tau^2 & \mathbf{c} \\ \mathbf{c}' & \mathbf{K}^*\end{bmatrix}\right)$$

$$\begin{aligned}\tau^2 &= \mathrm{Var}(\mu(s)) \\ \mathbf{c} &= \mathrm{Cov}(\mu(s), (\mu(s_1^*), \ldots, \mu(s_m^*))) \\ \mathbf{K}^* &= \mathrm{Cov}\left(\begin{bmatrix}\mu(s_1^*) \\ \vdots \\ \mu(s_m^*)\end{bmatrix}, \begin{bmatrix}\mu(s_1^*) \\ \vdots \\ \mu(s_m^*)\end{bmatrix}\right)\end{aligned}$$

Therefore,

$$\begin{aligned}
\tilde{\mu}(s) &= \mathrm{E}\left[\mu(s)|\mu(s_1^*,\ldots,s_m^*)\right] \\
&= 0 + c(s)K^{*-1}\begin{bmatrix}\mu(s_1^*)\\ \vdots \\ \mu(s_m^*)\end{bmatrix} \\
\mathbf{y} &= c(s)K^{*-1}\begin{bmatrix}\mu(s_1^*)\\ \vdots \\ \mu(s_m^*)\end{bmatrix} + \boldsymbol{\epsilon} \\
\begin{bmatrix}y_1\\ \vdots \\ y_n\end{bmatrix} &= \begin{bmatrix}c(s_1)\\ \vdots \\ c(s_n)\end{bmatrix}\mathbf{K}^{*-1}\begin{bmatrix}\mu(s_1^*)\\ \vdots \\ \mu(s_m^*)\end{bmatrix} + \boldsymbol{\epsilon} \\
&= \mathbf{H}\boldsymbol{\mu} + \boldsymbol{\epsilon} \\
\pi(-|\mathbf{y}) &\propto \mathrm{N}(\mathbf{y}|H\boldsymbol{\mu}, \sigma^2\mathbf{I}) \times \mathrm{N}(\boldsymbol{\mu}|\mathbf{0}, \mathbf{K}^*) \times \pi(-)
\end{aligned}$$

Now, integrating out $\boldsymbol{\mu}$,

$$\pi(-|\mathbf{y}) \propto \mathrm{N}(\mathbf{y}|\mathbf{0}, \sigma^2\mathbf{I} + H\mathbf{K}^*H') \times \pi(-)$$

So at each MCMC iteration, we'd need to invert $(\sigma^2\mathbf{I} + H\mathbf{K}^*H')$, which we can do using the Sherman-Woodbury-Morrison identity. Therefore, we have

$$(\sigma^2\mathbf{I} + H\mathbf{K}^*H')^{-1} = \frac{1}{\sigma^2}\mathbf{I} - \frac{1}{\sigma^2}H\left(\mathbf{K}^{*-1} + \frac{H'H}{\sigma^2}\right)^{-1}\frac{H'}{\sigma^2}$$

which, again, is $m \times m$ versus $n \times n$. This can all be done with a package in R called `spBayes`. For the homework, we will write our own code and compare it to the results from this package.

Note: None of these low rank models works when you have datasets with n greater than 20,000 or 30,000.

3 Sparsity-based Models

The idea is that when you are fitting a GP, you can make the covariance matrix of the likelihood sparse, i.e. induce zeros into the matrix following some structure to make the covariance easily invertible.

Gaussian Processes

AMS 268

February 24, 2016

Note:The second homework deadline has been extended to next Friday. The project submission deadline is the 15th of March and the presentations will take place March 7-11.

Recap:

$$y = \mu(s) + \epsilon,\ \mu \sim \text{GP}(0, \kappa(\cdot, \cdot))$$

For right now, assume the covariance kernel is

$$\begin{aligned}
\kappa(s, s') &= \tau^2 \exp\{-\phi||s - \mathcal{B}'||\} \\
y &= \tilde{\mu}(s) + \epsilon, \epsilon \sim \text{N}(0, \sigma^2) \\
s_1^*, \ldots, s_m^* &\rightarrow \text{ knot points} \\
\tilde{\mu}(s) &= E[\mu(s)|\mu(s_1^*), \ldots, \mu(s_m^*)] \\
&= \begin{bmatrix} c(s_1) \\ \vdots \\ c(s_n) \end{bmatrix} \mathbf{K}^{*-1} \begin{bmatrix} \mu(s_1^*) \\ \vdots \\ \mu(s_m^*) \end{bmatrix} \\
\mathbf{c} &= \text{Cov}(\mu(s), (\mu(s_1^*), \ldots, \mu(s_m^*)))
\end{aligned}$$

See the notes from the 22nd for more details. We can now abbreviate the model by saying that

$$\begin{aligned}
\mathbf{y} &= \mathbf{c}\mathbf{K}^{*-1}\boldsymbol{\mu}^* + \boldsymbol{\epsilon} \\
\boldsymbol{\epsilon} &\sim \text{N}(\mathbf{0}, \sigma^2\mathbf{I}) \\
\boldsymbol{\mu}^* &\sim \text{N}(\mathbf{0}, \mathbf{K}^*) \\
\mathbf{y}|\phi, \tau^2, \sigma^2 &\sim \text{N}(\mathbf{0}, \mathbf{c}\mathbf{K}^{*-1}\mathbf{c}' + \sigma^2\mathbf{I}) \\
\sigma^2 &\sim \text{IG}(a, b) \\
\tau^2 &\sim \text{IG}(c, d) \\
\phi &\sim \text{Unif}(a_1, a_2) \\
\text{So, } \pi(\phi, \tau^2, \sigma^2|\mathbf{y}) &\propto \text{N}(\mathbf{y}|\mathbf{0}, \mathbf{c}\mathbf{K}^{*-1}\mathbf{c}' + \sigma^2\mathbf{I}) \times \text{IG}(\sigma^2|a, b) \times \text{IG}(\tau^2|c, d) \times \text{Unif}(\phi|a_1, a_2)
\end{aligned}$$

Use `mcmc` package in R by Charles Geyer. We'll have to transform ϕ and multiply by the Jacobian, but otherwise, the package will do the Metropolis steps for you. Raj says it's a good idea to block update $\eta_1 = \log \frac{\phi - a_1}{a_2 - \phi}$, $\eta_2 = \log \sigma^2$, $\eta_3 = \log \tau^2$ together. Raj says that if we proceed without integrating out $\boldsymbol{\mu}^*$, then there will be Gibbs steps, but the mixing will be terrible. After you have posterior samples from the posterior, it is possible to draw $\boldsymbol{\mu}^*|\mathbf{y}$. Once we have that, we get $\boldsymbol{\mu}(s)$ because the relationship between $\boldsymbol{\mu}$ and $\boldsymbol{\mu}^*$ is deterministic.

Suppose you simulate $y = \mu(s) + \epsilon$, $\epsilon \sim \text{N}(0, \sigma^2)$ and $\mu \sim$ GP. You then simulate all ys in $\mathbf{y}$ from the multivariate normal. Then you fit a predictive process on that data. This would be unrealistic, as you would be using a non-stationary model to fit stationary data. That's one problem, but a more severe problem is

that your variance will be underestimated because we estimate $\mu(s)$ with $E[\mu(s)|\mu(s_1^*), \ldots, \mu(s_m^*)]$. This lead immediately to the **Modified Predictive Process (MPP)**. Even this is not the "best". Raj says the best is called **Nearest-Neighbor Gaussian Process (NNGP)**. This is super new, but we don't have the time to go into it. However, Raj says that he's working on something called **SMK**, which will be the "best".

So we're working with the predictive process

$$y = \tilde{\mu}(s) + \epsilon,\ \epsilon \sim \mathrm{N}(0, \sigma^2)$$

but the Modified Predictive Process is such that

$$y = \tilde{\mu}(s) + \eta(s) + \epsilon,\ \epsilon \sim \mathrm{N}(0, \sigma^2)$$

where $\eta(s_1), \ldots, \eta(s_n)$ are independent. So,

$$\mathrm{Var}(\mu(s)) = \tau^2 = \mathrm{Var}(\tilde{\mu}(s_i)) + \mathrm{Var}(\eta(s_i))$$

and

$$\eta(s) \overset{ind}{\sim} \mathrm{N}(\mathbf{0}, (\tau^2 - (\kappa(s, s_1^*), \ldots, \kappa(s, s_m^*))\mathbf{K}^{*-1}(\kappa(s, s_1^*), \ldots, \kappa(s, s_m^*))')$$

Raj then gave an outline on NNGP.

1 Sparsity-Based Gaussian Process

This method is essentially taking a space with two points, s_1 and s_2, and the covariance kernel is given as the same as before. (Gneting 2001) proposed a new class of covariance kernels, known as **compactly supported covariance kernels**. The idea is that the two points will have covariance greater than zero if they are within some threshold distance of each other. Otherwise, they will have covariance zero.

Sparsity-Based Gaussian Process

AMS 268

February 29, 2016

Covariance kernels result in a nonzero covariance for all components in the kernel. They can be small, but they're still nonzero. The idea behind sparsity based GP is that we set some distance such that beyond this threshold, the covariance between two components is zero. So we'll make K, the covariance kernel, sparse and employ a sparse matrix solve methods.

So we have Gaussian Process which has $O(n^3)$, and the Gaussian Predictive process has $O(m^3 + m^2 n) \approx O(n^2)$, so sparsity-based methods aim to beat this.

We can't induce sparsity by just chucking zeros into the covariance matrix randomly because we'll lose positive definiteness, and therefore, it will not be a valid covariance.

This lead to the idea of **compactly supported correlation functions**. This presents a correlation kernel $K(s, s', \delta)$ such that $K(s, s', \delta) = 0$ if $||s - s'|| > \delta$. For example,

$$K(s, s', \delta) = \left(1 - \frac{||s - s'||}{\delta}\right)_+^4 \left(1 + 4\frac{||s - s'||}{\delta}\right)$$
$$K(s, s', \delta) = \left(1 - \frac{||s - s'||}{\delta}\right)_+^2 \left(1 + 2\frac{||s - s'||}{\delta}\right)$$

To prove that some covariance kernel is valid, you have to prove that it satisfies the Kolmogorov consistency theorem.

Supposed you have a GP $\mu \sim \text{GP}(\cdot, \kappa(\cdot, \cdot, \boldsymbol{\theta}))$. So before, we were fitting $y = \mu(s) + \epsilon$ to the data, but now we'll fit the model $y = \mu(s)w(s) + \epsilon$ to the data, where

$$w \sim \text{GP}(\mathbf{0}, \kappa_\delta(\cdot, \cdot))$$

and κ_δ is a compactly-supported correlation function. Now let's define K_1 as the $n \times n$ covariance matrix such that

$$K_{1,ij} = \text{Cov}(\mu(s_i)w(s_i), \mu(s_j)w(s_j))$$

Now we have the joint posterior

$$\pi(-|\mathbf{y}) \propto \text{N}(\mathbf{y}|\mathbf{0}, K_1 + \sigma^2\mathbf{I}) \times \pi(-)$$

So what this means is that

$$\begin{aligned} K_{1,ij} &= \text{Cov}(\mu(s_i)w(s_i), \mu(s_j)w(s_j)) \\ &= \text{Cov}(\mu(s_i), \mu(s_j))\text{Cov}(w(s_i), w(s_j)) \\ &= \kappa(s_i, s_j, \boldsymbol{\theta})\kappa_\delta(s_i, s_j) \end{aligned}$$

because μ and w are independent processes. Keep in mind that δ would be a tuning parameter here, which you would have to choose somehow. If you'd like to induce more sparsity in K_1, choose a smaller value for

δ.

So Sang (2012) suggested that we change this model

$$y = \mu(s)w(s) + \epsilon(s)$$

and instead look at this model.

$$y = \mathrm{E}(\mu(s)|\mu(s_1^*), \mu(s_2^*), \ldots, \mu(s_m^*)) + (\mu(s) - \mathrm{E}(\mu(s)|\mu(s_1^*), \mu(s_2^*), \ldots, \mu(s_m^*)))w(s) + \epsilon(s)$$

This was suggested because the sparsity-based model overestimates the variance due to the product of the GPs. This model improves the fit of the model significantly, but it does increase the complexity of the model significantly. This is done because $\mathrm{E}(\mu(s)|\mu(s_1^*), \mu(s_2^*), \ldots, \mu(s_m^*))$ will capture the long range dependence between observations. However, it produces an overly-smooth process, which may miss some local dependence. So we add $(\mu(s) - \mathrm{E}(\mu(s)|\mu(s_1^*), \mu(s_2^*), \ldots, \mu(s_m^*)))w(s)$ to capture that local structure.

Fuentes Montse and Michael Stein did a bunch of work on this around 2009, but not much has been done on it since. Suppose that the covariance kernel is **isotropic**, that is, it depends solely on the distance between knots, i.e. $K(s_i, s_j) = K(||s_i - s_j||)$. Let's say

$$\begin{aligned} d &= ||s - s'|| \\ K(d) &= \int e^{-2\pi i \lambda} \psi(\lambda) d\lambda \\ \psi(\lambda) &= \text{ Frourier transform of } K \\ \text{For the Matérn correlation } (\phi, \nu),\ \psi(\lambda) &= \frac{1}{\left[1 + \frac{||\lambda||}{\phi}\right]^{\nu+1}} \end{aligned}$$

Broadly, the idea is that instead of staying in the data domain, work in the Fourier domain. You then use some cleverness to estimate $\psi(\lambda)$ efficiently, and then use the inverse Fourier transform.

Variational Inference

AMS 268

March 2, 2016

I'm presenting on March 9th!

We use variational inference when the posterior distribution is difficult to fit directly in a mode-based way, and no model-based approximation seems to be available. The main idea is to start by approximating the posterior with something easy. The differences between ADF and VB are that in ADF you specify the approximating class, but you don't do this in general in VB.

One setting in which this can happen is if you have observations y_i with corresponding latent variables z_i. The posterior is nasty, and we'd need to approximate it. We could approximate $p(\mathbf{z}|\mathbf{y})$ by $q(\mathbf{z})$, where q minimizes

$$KL(p||q) = E\left[\log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{y})}\right]$$

We do this by first recognizing that

$$\log(p(\mathbf{y})) = \log\left(E_q\left[\frac{p(\mathbf{y},\mathbf{z})}{q(\mathbf{z})}\right]\right) \geq E_q\left[\log \frac{p(\mathbf{y},\mathbf{z})}{q(\mathbf{z})}\right]$$

(the inequality comes from Jensen's inequality) and then attempting to maximize the last term. This term is known as the ELBO (Evidence Lower Bound), and we want to make this ELBO as close as possible to the second term. What does this have to do with the KL divergence of the posterior? If we work through the equations, then we see that

$$KL(p||q) + ELBO = \log p(\mathbf{y})$$

So we minimize KL by maximizing ELBO because $\log(p(\mathbf{y}))$ is a constant.

1 Mean Field Variational Approximation

This is the most-used form of VB. Statisticians and machine learners stole it from physics. In this case, we assume that $q(z_1, \ldots, z_m) = q_1(z_1) \cdots q_m(z_m)$. We can estimate the q_js by using a result from variational calculus

$$q_j^*(z_j) \propto \exp\left(E_{Z_{-j}}[\log p(\mathbf{z}, \mathbf{y})]\right)$$

The closed-form for this is available only for the mean field approximation. Note that you can "block" the correlated latent variables together using VB, if you'd like
(i.e. $q(z_1, \ldots, z_m) = q(z_1, z_5, z_{10})q(z_2, z_3, z_4)q(z_6, z_7, z_8, z_9)q(z_{11}, \ldots, z_m)$)
Raj showed an example of this using Gaussian mixtures. In the case of his example, there were two variables that needed to be simulated in a posterior, so there will be an iterative convergence algorithm to find optimal values for the parameters in q.

Variational Inference - Gaussian Process Latent Variable Models

AMS 268

March 4, 2016

These are models of the form

$$y = \mu(\mathbf{x}) + \epsilon$$

You have latent variables, but you don't know which ones are affecting the observations (responses). This is the nonparametric analogue of PCA. All of this content will be coming from Michael Titsias (2009). We have the observed responses $\mathbf{y}$ which is an $N \times D$ matrix. We're also assuming that each element in an individual observation of $\mathbf{y}$ is conditionally independent. The predictors $\mathbf{x}$ is $N \times Q$ where Q is the number of latent variables and Q is the number of latent variables such that $Q \ll D$. So then

$$P(\mathbf{y}|\mathbf{x}) = \prod_{d=1}^{D} p(\mathbf{y}_d|\mathbf{x})$$

Suppose that the correlation kernel is

$$K(x, x') = \sigma_f^2 \exp\left\{ -\frac{1}{2} \sum_{q=1}^{Q} \alpha_q (x_q - x_q')^2 \right\}$$

The part in the exponential is known as the GP-ARD kernel. We also have that the latent variables that are distributed as

$$p(\mathbf{x}) = \prod_{n=1}^{N} \mathrm{N}(\mathbf{x}_n|\mathbf{0}, \mathbf{I}_Q)$$

We can write the joint distribution

$$p(\mathbf{y}, \mathbf{x}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$$

Note: If machine learners refer to a sparse Gaussian Process, they are talking about a Gaussian Predictive Process.
Getting the posterior distribution $p(\mathbf{x}|\mathbf{y})$ is difficult, so the first stage in approximating it using variational methods comes as

$$q(\mathbf{x}) = \prod_{n=1}^{N} \mathrm{N}(\mathbf{x}_n|\boldsymbol{\mu}_n, S_n)$$

where S_n is a diagonal matrix. Now the next thing to do is to look at the ELBO, which was discussed last

class.

$$\begin{aligned}
ELBO &= \int q(\mathbf{x}) \log \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{q(\mathbf{x})} dx \\
&= \int q(\mathbf{x}) \log p(\mathbf{y}|\mathbf{x}) - \int q(\mathbf{x}) \log \frac{q(\mathbf{x})}{p(\mathbf{x})} dx \\
&= \sum_{d=1}^{D} \int q(\mathbf{x}) \log p(\mathbf{y}_d|\mathbf{x}) dx - \int q(\mathbf{x}) \log \frac{q(\mathbf{x})}{p(\mathbf{x})} dx \\
&= \tilde{F}(q) - KL(q||p) \\
\tilde{F}(q) &= \sum_{d=1}^{D} \int q(\mathbf{x}) \log p(\mathbf{y}_d|\mathbf{x}) dx \\
&= \sum_{d=1}^{D} \tilde{F}_d(q) \\
p(\mathbf{y}_d, f_d, u_d|\mathbf{x}, \mathbf{z}) &= p(y_d|f_d)p(f_d|u_d, \mathbf{x}, \mathbf{z})p(u_d|\mathbf{z})
\end{aligned}$$

where

$$f_d = \begin{pmatrix} \mu(\mathbf{x}_1) \\ \vdots \\ \mu(\mathbf{x}_N) \end{pmatrix} \qquad u_d = \begin{pmatrix} \mu(x_1^*) \\ \vdots \\ \mu(x_M^*) \end{pmatrix} \qquad \mathbf{z} = \{x_1^*, \ldots, x_M^*\}$$

and

$$\begin{aligned}
p(y_d|f_d) &= \mathrm{N}(y_d|f_d, \boldsymbol{\beta}^{-1}\mathbf{I}_N) \\
p(f_d|u_d, \mathbf{x}, \mathbf{z}) &= \mathrm{N}(f_d|K_{NM}K_{MM}^{-1}u_d, K_{NN} - K_{NM}K_{MM}^{-1}K_{MN}) \\
p(u_d|\mathbf{z}) &= \mathrm{N}(u_d|\mathbf{0}, K_{MM})
\end{aligned}$$

The second stage of the variational approximation is that $p(f_d, u_d|y_d, \mathbf{x})$ is approximated by $q(f_d, u_d) = p(f_d|u_d, \mathbf{x})\phi(u_d)$. Using the ELBO, we find that

$$\begin{aligned}
p(y_d|\mathbf{x}) &\geq \int \phi(u_d) \log \frac{pp(u_d)\mathrm{N}(y_d|K_{NM}K_{MM}^{-1}u_d, \boldsymbol{\beta}^{-1}\mathbf{I}_N)}{\phi(u_d)} du_d \\
&\quad - \frac{\boldsymbol{\beta}}{2}\mathrm{Tr}(K_{NN} - K_{NM}K_{MM}^{-1}K_{MN}) \\
\tilde{F}_d(q) &\geq \int q(\mathbf{x}) \left[\int \phi(u_d) \log \frac{p(u_d)\mathrm{N}(y_d|K_{NM}K_{MM}^{-1}u_d, \boldsymbol{\beta}^{-1}\mathbf{I}_N)}{\phi(u_d)} du_d - \frac{\boldsymbol{\beta}}{2}\mathrm{Tr}(K_{NN}) + \frac{\boldsymbol{\beta}}{2}\mathrm{Tr}(K_{MM}^{-1}K_{MN}K_{NM}) \right] d\mathbf{x} \\
&= \int \phi(u_d) \left[E_{q(\mathbf{x})} \left[\log \mathrm{N}(y_d|K_{NM}K_{MM}^{-1}u_d, \boldsymbol{\beta}^{-1}\mathbf{I}_N) \right] + \log \frac{p(u_d)}{q(u_d)} \right] du_d \\
&\quad - \frac{\boldsymbol{\beta}}{2}\mathrm{Tr}\left(E_{q(\mathbf{x})}[K_{MN}] \right) + \frac{\boldsymbol{\beta}}{2}\mathrm{Tr}\left(K_{MM}^{-1} E_{q(\mathbf{x})}\left[K_{MN}K_{NM} \right] \right)
\end{aligned}$$

Now define the quantities

$$\begin{aligned}
\psi_0 &= \mathrm{Tr}\left(E_{q(\mathbf{x})}[K_{NN}] \right) \\
\psi_1 &= E_{q(\mathbf{x})}[K_{NM}] \\
\psi_2 &= E_{q(\mathbf{x})}[K_{MN}K_{NM}]
\end{aligned}$$

where

$$(\psi_1)_{nm} = \sigma_f^2 \prod_{q=1}^{Q} \frac{e^{-\frac{1}{2}\frac{\alpha_q(\boldsymbol{\mu}_{nq} - \mathbf{z}_{mq})^2}{\alpha_q S_{nq} + 1}}}{(\alpha_q S_{nq} + 1)^{1/2}}$$

So then

$$\tilde{F}_d(q) \geq \log\left[\frac{\boldsymbol{\beta}^{N/2}|K_{MM}|^{1/2}}{(2\pi)^{N/2}|\boldsymbol{\beta}\psi_2 + K_{MM}|^{1/2}} e^{-\frac{1}{2}\mathbf{y}_d' W \mathbf{y}_d}\right] - \boldsymbol{\beta}\frac{\psi_0}{2} + \frac{\boldsymbol{\beta}}{2}\mathrm{Tr}(K_{MM}^{-1}\psi_2)$$
$$W = \boldsymbol{\beta}\mathbf{I}_N - \boldsymbol{\beta}^2\psi_1(\boldsymbol{\beta}\psi_2 + K_{MM})^{-1}\psi_1^T$$

Now we have

$$q(\{u_d, f_d\}_{d=1}^D, \mathbf{x}) = \left[\prod_{d=1}^D \{p(f_d|u_d, \mathbf{x})\phi(u_d)\}\right] q(\mathbf{x})$$

Our goal was to approximate $\pi(\{f_d, u_d\}_{d=1}^D, \mathbf{x}|\mathbf{y})$, so now we have a way to do that.

Stochastic Gradient Descent

AMS 268

March 11, 2016

This is an optimization method used for "big data". It's not very recent, as it was first proposed by RObbins and Monroe in 1954. Suppose you have

$$\pi(\theta|x_1,\ldots,x_n) \propto \left[\prod_{i=1}^{n} p(X_i|\theta)\right] p(\theta)$$

So the log posterior is

$$\pi(\theta|x_1,\ldots,x_n) = \sum_{i=1}^{n} \log(p(x_i|\theta)) + \log p(\theta) + C$$

Maximizing $\pi(\theta|x_1,\ldots,x_n)$ w.r.t. θ gives

$$\begin{aligned}\Delta\theta &= \theta_{t+1} - \theta_t \\ &= \epsilon\left(\triangledown \log(p(\theta_t)) + \sum_{i=1}^{n} \triangledown \log(p(x_i|\theta_t))\right)\end{aligned}$$

For every iteration, you take a subsample $\{y_1,\ldots,y_N\}$ where $N \ll n$. The idea is that

$$\frac{1}{n}\sum_{i=1}^{n} \log(p(x_i|\theta)$$

will be estimated unbiasedly by

$$\frac{1}{N}\sum_{j=1}^{N} \log(p(x_j|\theta)$$

So then

$$\begin{aligned}\Delta\theta &= \theta_{t+1} - \theta_t \\ &= \epsilon_t\left(\triangledown \log(p(\theta_t)) + \sum_{i=1}^{n} \triangledown \log(p(y_i|\theta_t))\right) \\ &\approx \epsilon_t\left(\triangledown \log(p(\theta_t)) + \frac{n}{N}\sum_{j=1}^{N} \triangledown \log(p(y_j|\theta_t))\right)\end{aligned}$$

Which has the properties $\sum_{t=1}^{\infty} \epsilon_t^2 < \infty$ and $\sum_{t=1}^{\infty} \epsilon_t = \infty$. This will result in a point estimate for θ. Is there any way that we can use this to take samples from the posterior distribution for big t? This would solve the issue of big data by having cluster sizes of the data be manageably small. So Welling & Teh proposed using the Stochastic-Langevin methods in 2011. What is Stochastic-Langevin?
If we use the algorithm

$$\theta_{t+1} - \theta_t = \epsilon_t\left[\triangledown \log p(\theta_t) + \frac{n}{N}\sum_{j=1}^{n} \triangledown \log p(y_j|\theta_t)\right] + \eta_t,\ \eta_t \sim \mathrm{N}(0,\epsilon_t),$$

good things will happen.

1. Suppose $\{\theta_t\}_t$ are drawn as a proposed candidate for θ in the Metropolis step.
2. Welling & Teh (2011) intuitively argued that as $t \to \infty$, the Metropolis acceptance ratio $\to 1$
3. This implies we do not need to do Metropolis at all
4. Note that this method should *only* be used for big n