10/2/19

Physics 8805: Learning from Data: Lecture 12

On board: Reminder: mini-project I due Friday. Questions?

Notebooks for today:

- MCMC – diagnostics.ipynb
- correlation_intuition.ipynb (if needed – or do from scratch)
- dealing_with_outliers.ipynb

Return to MCMC diagnostics

- Reminder: "Key purpose of MCMC is not to explore the posterior but to estimate expectation values."
- Recap convergence issues and strategies on (66)
- First look at Gelman-Rubin diagnostic on (67)
  ⇒ new version from March 2016: arXiv: 1903.08008

- Figures to make every time you run MCMC (Hogg and Foreman-Mackey)
  - Trace plots → burn-in length can be seen, identify problems with model or sampler, qualitative judge of convergence.
    - Use convergence diagnostic such as Gelman-Rubin

- Corner plots – If D-dimensional parameter space, plot all D and all $\binom{D}{2}$ histograms to show low-level covariances and non-linearities. "...They are remarkable for locating expected and unexpected parameter relationships, and often invaluable for suggesting re-parameterizations and transformations that simplify your problem."

- Posterior predictive plots
  - Take K random samples from your chain, plot the prediction each sample makes for the data and overplot the observed data. "This plot gives a qualitative sense of how well the model fits the data and it can identify problems with sampling or convergence."

see arXiv: 1710.06068 sect 9 for "Troubleshooting and advice"

10/2/19

- Building intuition about correlations

- Recall that we parametrize a $N=2$ parameter covariance matrix as

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}$$

- Use correlation_intuition.ipynb to develop our intuition about what this implies for a multi-variate normal distribution (in this case bivariate):

$$\vec{x} \mid \vec{\mu}, \Sigma \sim N(\vec{\mu}, \Sigma) \Rightarrow p(\vec{x}) = \frac{1}{\sqrt{(2\pi)^N |\Sigma|}} e^{-(\vec{x}-\vec{\mu})^T \Sigma^{-1} (\vec{x}-\vec{\mu})}$$

($|\Sigma|$: determinant)

and also to look at how we can implement the linear algebra in Python.

- Note that if we represent a vector as a numpy array,

```
x_vec = np.array([x0, x1, x2])
```

we can do dot products and multiply matrices, but the difference between a row vector and column vector is only built in when we represent the vector as an $N \times 1$ or $1 \times N$ matrix.

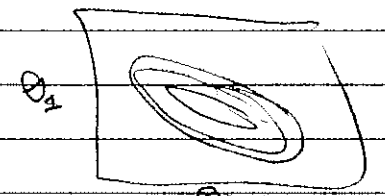
** NOTE: Do not use the numpy.matrix class (deprecated!)

- Try the examples so the association of σ_1, σ_2, and ρ with the shape of the contour becomes intuitive.

10/2/19

- Sampling from correlated distributions: What to do?

- If our posterior has projections that look like:



and we are doing Metropolis-Hastings (MH) sampling, how do we decide on a step size?

- The problem is that we want to step differently in different directions: A "long enough" step size to explore the long axis will lead to many rejections in the orthogonal direction.

  $\Rightarrow$ we do not want an isotropic step proposal!

- If we propose steps $p(\vec{x}) = \frac{1}{\sqrt{(2\pi)^N |\Sigma|}} e^{-\vec{x}^T \Sigma^{-1} \vec{x}}$, then we don't need to take $\Sigma \propto \sigma^2 \mathbb{1}_N$! We have $N(N-1)$ parameters to "tune" to reduce the correlation time.
  - This is increasingly difficult as N increases.

- One improvement is to do a linear transformation of $\vec{\theta} \to \vec{\theta}' = A\vec{\theta} + \vec{B}$ such that $\vec{\theta}'$ is uncorrelated with similar σ_i's 

- Or, we can use an "affine invariant" sampler like emcee.
  - An affine transformation is an invertible mapping from $\mathbb{R}^N \to \mathbb{R}^N$.
    $\Rightarrow \vec{y} = A\vec{x} + \vec{B}$ $\Rightarrow$ combination of stretching, rotation, translation.
  - affine invariant means that the sampler performs equally well on all affine transformations of a distribution.
  - So emcee figures out how to make the appropriate steps.
  - It does this by using the many walkers at time t, which have sampled the space, to construct an appropriate affine compatible update step for $t+1$. If time permits, we'll examine this further later.

10/2/19

Why Bayes is better II: dealing with outliers

- Our exploration of different approaches to handling outliers is worked out in dealing_with_outliers.ipynb.
  - The details are in the notebook; here we give an overview.

- Our example is linear regression with data outliers, meaning we fit a linear function (here just a line in one variable) to data that includes one or more points that are many standard deviations from the trend.

- The model setup is familiar $y_{exp} = y_m + \delta y_{exp}$ (y_m; normal with σ_0 for a points)

$$\Rightarrow p(x_i, y_i | \theta, \sigma_0) = \frac{1}{\sqrt{2\pi}\sigma_0} e^{-(y_i - y_m(x_i;\theta))^2 / 2\sigma_0^2} \qquad y_m(x) = \theta_1 x + \theta_0$$

- So we can define a likelihood. Later, when we'll need priors, we will take them as uniform (even though not well motivated!)

Frequentist: Standard likelihood approach and Huber loss
- The former shows the out-sized influence of outliers with a squared loss function,
- The Huber loss switches to a linear loss function for larger deviations (with the cross over parametrized), which reduces the loss contribution of outliers, ⇒ much more intuitive
- Some issues are identified!

Bayesian approaches
1. conservative model
2. good-and-bad data model
3. Cauchy formulation
4. Many nuisance parameters

} step through and discuss how these work in each case.
Sivia chapter 8 has further commentary!