Data Science Companion

Greg Simon, gregorygsimon@gmail.com

November 7, 2020

Abstract

A reference for basic data science tools and vocabulary, explaining essential terms and concepts, examining core ideas in major areas, and putting methods in context. Includes relevent keywords and references.

Contents

1 Bayesian Statistics

We follow [Fon19]. Frequentist hypothesis testing has some flaws.

- We decide on a statistical test, null hypothesis, and significance level subjectively, often based on tradition
- The null hypothesis is often clearly false with enough data.
- Easy to misinterpret the results, e.g. p-values.

Instead of testing, we want **estimation** to measure *how different* two groups are and we want to include a an estimate of the *uncertainty*, both due to our lack of knowledge of model parameters (*'epistemic uncertainty'*) as well as uncertainty due to stochasticity of the system (*aleatory uncertainty*).

The Bayesian approach stems from the equation:

$$\underbrace{\mathbb{P}(\theta|y)}_{\text{posterior}} = \frac{\overbrace{\mathbb{P}(y|\theta)}^{\text{likelihood}} \cdot \overbrace{\mathbb{P}(\theta)}^{\text{prior}}}{\underbrace{\mathbb{P}(y)}_{\text{marginal likelihood}}}$$

By inputing in prior probability distributions, we get posterior distributions out. The general steps are as follows:

1. **Specify a probability model** – assign distributions for unknown parameters, data, covariates, etc.
2. **Calculate the posterior distributions** – difficult but there are many different methods for this.
3. **Check your model** – does it fit the data? Are the conclusions reasonable? Are the outputs sensitive to changes in the model structure?
4. **Use the posterior distribution to get what you want** – point estimates, credible intervales, quantiles, predictions, etc.

One of the main challenges is calculating.

1.1 Markov Chain Monte Carlo (MCMC)

Recall $\mathbb{P}(\theta|y) \propto \mathbb{P}(y|\theta)\mathbb{P}(\theta)$ and we can calculate the right side. We don't know the normalizing constant $\mathbb{P}(y) = \int_\theta P(y|\theta)\mathbb{P}(\theta)\, d\theta$.

Good explanation from [jpi11]

> The goal of MCMC is to draw samples from the probability distribution on the right without having to know its exact height at any point. The way MCMC achieves this is to "wander around" on that distribution in such a way that the amount of time spent in each location is proportional to the height of the distribution.
>
> The simplest variant of the Metropolis-Hastings algorithm (independence chain sampling) achieves this as follows: assume that in every (discrete) time-step, we pick a random new "proposed" location (selected uniformly

across the entire surface). If the proposed location is higher than where we're standing now, move to it. If the proposed location is lower, then move to the new location with probability p, where p is the ratio of the height of that point to the height of the current location. (i.e., flip a coin with a probability p of getting heads; if it comes up heads, move to the new location; if it comes up tails, stay where we are). Keep a list of the locations you've been at on every time step, and that list will (asyptotically) have the right proportion of time spent in each part of the surface. (And for the A and B hills described above, you'll end up with twice the probability of moving from B to A as you have of moving from A to B).

There are more complicated schemes for proposing new locations and the rules for accepting them, but the basic idea is still: (1) pick a new "proposed" location; (2) figure out how much higher or lower that location is compared to your current location; (3) probabilistically stay put or move to that location in a way that respects the overall goal of spending time proportional to height of the location.

Good breakdown of MCMC in PyMC3 and of a simple change point model is in [Pil16, Ch. 1].

1.1.1 Metropolis-Hastings algorithm

1.1.2 Gibbs sampling

1.1.3 Hamiltonian Monte Carlo

Random-walk sampling (like Metropolis-Hastings) is not efficient in high dimension. We want an algorithm that samples from the area of the parameter space that contains most of the non-zero probability. This region is called the *typical set*.

The Hamiltonian Monte Carlo (HMC) avoids random walk behavior by simulating a physical system governed by Hamiltonian dynamics. The math is explained well in the following report [Nea93].

We start with a state $\chi = (s, \phi)$ with a position s and velocity ϕ. As in physics, the joint probability is proportional to an invariant Hamiltonian function:

$$\mathbb{P}((s,\phi)) \propto \exp(-\mathcal{H}(s,\phi))$$

where the Hamiltonian is the sum of the two types of energy:

$$\mathcal{H}(s,\phi) = \underbrace{E(s)}_{\text{potential}} + \underbrace{K(\theta)}_{\text{kinetic}} = E(s) + \frac{1}{2}\sum_i \phi_i^2.$$

These satisfy the differential equations:

$$\frac{ds_i}{dt} = \frac{\partial \mathcal{H}}{\partial \phi_i} = \phi_i$$
$$\frac{d\phi_i}{dt} = -\frac{\partial \mathcal{H}}{\partial s_i} = -\frac{\partial E}{\partial s_i}$$

Often we assume that $\mathbb{P}(\phi)$ is often taken to be the univariate Gaussian. So we can use a **leap-frog** time discretization of this differential equation. Care must be maintained to preserve *volume conservation* and *time reversibility*. This performs half-step updates to the velocity at time $t + \epsilon/2$ which is used to compute $s(t + \epsilon)$ and $\phi(t + \epsilon)$. We also introduce an accept/reject stage with some probability to correct for the bias of discretization as well as floating-point rounding errors.

1.1.4 No-U-Turn Sampler (NUTS)

Summarized as "Adaptively Setting Path Lengths in Hamiltonian Monte Carlo" [HG14]. Auto-tunes number of steps L and step size ϵ. Uses a recursive algorithm to build a set of likely candidate points, stopping automatically if it begins to retrace its steps.

1.2 Model Checking

Two types: **Convergence diagnostics** and **Goodness of fit**. The former is intended to detect lack of convergence in MCMC sample - e.g. to ensure you have not halted your sampling too early.

A converged model is not guaranteed to be a good model. Goodness of fit is used to check the internal validity of the odel by comparing predictions from the model to the data used to fit the model.

Informally, you can plot and inspect the traces and histograms of the observed MCMC sample (as in `arviz.plot_trace()`). The trace may clearly not yet be convergent, or it may appear to be stationary about its mean.

Another informal method is to initialize with differenting starting values. If each trace converges towards the same equilibrium, this is evidence for *ergodicity*[1]. Carefull for *metastability* – characterized by stability for a long duration but then moving to another region of the parameter space.

1.2.1 Gelman-Rubin diagnostic

Gelman-Rubin diagnostic uses multiple chains to check for convergence, based on the idea that if multiple chains have converged then they should appear very similar to one-another. It uses an analysis of variance approach, calculating between-chain variance B and a within-chain variance W and assess whether they are different enough to worry about lack of convergence.

This allows us to estimate the *marginal posterior variance* of the parameter of interest and defines an $\hat{R}$ statistics that should be close to 1 if the chains are convergent.

$$\widehat{\mathrm{Var}}(\theta|y) = \frac{n-1}{n}W + \frac{1}{n}B$$

$$\hat{R} = \sqrt{\frac{\widehat{\mathrm{Var}}(\theta|y)}{W}}$$

A great Q&A and common misunderstandings for MCMC is summarized in [RC20].

[1] ergodicity is the tendency for some Markov chains to converge to the true unknown value from diverse starting states

1.3 References

Fonnesbeck, Chris (2019). "An introduction to Markov Chain Monte Carlo using PyMC3". In: *PyData*. URL: https://www.youtube.com/watch?v=SS_pqgFziAg.

Hoffman, Matthew D. and Andrew Gelman (2014). "The No-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo". In: *J. Mach. Learn. Res.* 15.1, pp. 1593–1623. URL: http://www.stat.columbia.edu/~gelman/research/published/nuts.pdf.

jpillow, stackexchange (2011). *How would you explain Markov Chain Monte Carlo (MCMC) to a layperson?* Cross Validated. URL: https://stats.stackexchange.com/q/12657.

Neal, Radford M. (1993). *Probabilistic Inference Using Markov Chain Monte Carlo Methods*. Tech. rep. Unpublished Report. URL: http://www.cs.toronto.edu/~radford/ftp/review.pdf.

Pilon, Cameron (2016). *Bayesian methods for hackers : probabilistic programming and Bayesian inference*. New York: Addison-Wesley. URL: https://github.com/CamDavidsonPilon/Probabilistic-Programming-and-Bayesian-Methods-for-Hackers.

Robert, Christian P. and Wu Changye (2020). *Markov Chain Monte Carlo Methods, a survey with some frequent misunderstandings*. arXiv: 2001.06249.

2 General Machine Learning Concepts

2.1 Model Selection

For a model S, the *prediction risk* is defined to be

$$R(S) = \sum_{i=1}^{n} \mathbb{E}[(\hat{Y}_i(S) - Y_i*)]$$

where $\hat{Y}_i$ are the predicted values and Y_i* are the values of a future observation at covariate values X_i. The *training error* is

$$\hat{R}_{tr}(S) = \sum_{i=1}^{n} \left(\hat{Y}_i(S) - Y_i\right)^2$$

The training error is a downward biased estimator for the prediction risk, and

$$\text{optimism}(S) = \text{bias}\left(\hat{R}_{tr}(S)\right) = \mathbb{E}(\hat{R}_{tr}(S)) - R(S) = -2\sum_{i=1}^{n} \text{Cov}(\hat{Y}_i, Y_i)$$

This leads to **Mallow's C_p statistic** defined by:

$$\hat{R}(S) = \hat{R}_{tr}(S) + 2d\sigma_\epsilon^2$$

where σ_ϵ^2 is the estimate of the standard deviation of the models error and d is the number of free inputs or basis functions in the model. [HTF01, §7.4].

2.1.1 Akaike information criterion (AIC)

The *Akaike Information Criterion* is (proportional to)

$$\text{loglik}_{\text{MLE}} - |S|$$

The log-likelihood of the model at the MLE minus the dimension of free parameters in the model.

We start with a set of models $\{M_1, M_2 \ldots\}$. Let $\widehat{f_j}(x) := \hat{f}(x, \widehat{\beta_j})$ be the estimated probability function obtained by using the parameters β_j that realize the maximimum likelihood estimate for model M_j.

One approach [Was13, §13.9] to consider the *Kullback-Leibler distance* defined by:

$$D(f, g) = \sum_x f(x) \log\left(\frac{f(x)}{\hat{f}(x)}\right)$$

And specifically consider the risk function $R(f, \hat{f}) = \mathbb{E}(D(f, \hat{f}))$. We can write

$$D(f, g) = sum_x f(x) \log f(x) - \sum_x f(x) \log \hat{f}(x)$$

So finding $\hat{f}$ that minimizes risk is equivalent to minimizing $a(f, \hat{f}) := \mathbb{E}\left(\sum_x f(x) \log \hat{f}(x)\right)$. The AIC is an appoximately unbiased estimate of $a(f, \hat{f})$.

2.1.2 Bayes information criterion (BIC)

With models with a set of models $M_1, M_2, \ldots$ and observed data Z, we have

$$\frac{\mathbb{P}(M_m|Z)}{\mathbb{P}(M_\ell|Z)} = \frac{\mathbb{P}(M_m)}{\mathbb{P}(M_\ell)} \cdot \frac{\mathbb{P}(Z|M_m)}{\mathbb{P}(Z|M_\ell)}$$

The rightmost factor is what we are concerned with, the *Bayes Factor*:

$$\mathrm{BF}(Z) = \frac{\mathbb{P}(Z|M_m)}{\mathbb{P}(Z|M_\ell)}$$

We seek to approximate the integral $\mathbb{P}(Z|M_m) = \int \mathbb{P}(Z|\theta, M_m)\mathbb{P}(\theta|M_m)d\theta$, (where the integral is over the space of parameters θ). This can be approximated using the MLE $\hat{\theta}_m$ for M_m

$$\log \mathbb{P}(Z|M_m) \approx \log \mathbb{P}(Z|\hat{\theta}_m, M_m) - \frac{d_m}{2} \log N + O(1)$$

So if we take our loss function to be $-2 \log \mathbb{P}(Z|\hat{\theta}_m, M_m)$ then we reach the BIC criterion with the goal of minimizing:

$$\mathrm{BIC} = -2\mathrm{loglik} + (\log N) \cdot d$$

Another great resource for BIC is [Raf95].

2.2 Fischer information

2.3 VC dimension

In a simple case, consider two class classification problem where $f(\mathbf{x}, \alpha) \in \{-1, 1\}$ for some parameter α. A given set of ℓ points can be labeled in 2^ℓ possible ways. If there is a fucntion $f(\cdot, \alpha)$ which correctly classifies all ℓ points, then we say these points is *shattered* by the set $\{f(\cdot, \alpha)\}$. The VC dimension is the maximum number of training points that can be shattered by $\{f(\alpha)\}$. If the VC dimension is h, then there is at least one set of points h that can be shattered but not *every* set of h points will be shattered. [Bur04, §2.1]

2.4 Ensemble methods

2.4.1 Bagging

2.4.2 Boosting

2.5 References

Burges, Christopher (2004). "A Tutorial on Support Vector Machines for Pattern Recognition". In: *Data Mining and Knowledge Discovery* 2, pp. 121–167.

Efron, Bradley and Trevor Hastie (2016). *Computer Age Statistical Inference: Algorithms, Evidence, and Data Science*. 1st. USA: Cambridge University Press.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc.

Raftery, Adrian E. (1995). "Bayesian Model Selection in Social Research". In: *Sociological Methodology* 25, pp. 111–163. URL: http://www.jstor.org/stable/271063.

Wasserman, L. (2013). *All of Statistics: A Concise Course in Statistical Inference*. Springer Texts in Statistics. Springer New York.

3 Regression

3.1 Regularization

3.1.1 Ridge (L^2)

$$SSE_{L2} = \sum_i (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{P} \beta_j^2$$

3.1.2 Lasso (L^1)

$$SSE_{L1} = \sum_i (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{P} |\beta_j|$$

Tends to shrink coefficients to 0 and almost performs feature selection.

3.2 Decision trees

Mostly following the notation in [HTF01, §9.2], our aim is to split into M regions $R_1, \ldots, R_M$ where we model the response as a constant c_i in each region $f(x) = \sum_{i=1}^{M} c_i I(x \in R_i)$. We inductively want to decide if and where to further split each region.

To do this, we seek splitting variable j and split point s that partition space into two half planes $L(j,s)\{X|X_j \leq s\}$ and $R(k,s) = \{X|X_j > s\}$ where we achieve the minimum of:

$$\min_{j,s} [\text{cost}(L) + \text{cost}(R)]$$

The cost function for regression is generally the sum-of-squared errors taking the average of each region as the predicted value of all points in that region.

To avoid overfitting, we may add all cost functions over the entire tree as well as an overall penalty to the cost function of the form $\alpha|T|$ where $|T|$ is the number of terminal nodes in T.

3.3 Classification Trees

For classification, we want other cost functions. Suppose we have labels $1, \ldots, K$. In region R_j, define the observed probability of k to be $\hat{p}_{jk}$.

Gini impurity [HTF01, §9.2.3] For classification, the *Gini impurity* measures the probability that an element would be labeled incorrectly. This is:

$$\sum_k \mathbb{P}(\text{actually } k)\mathbb{P}(\text{predicted } k|\text{actual } k) = \sum_k \hat{p}_{jk}$$

3.3.1 Oblique decision trees

Using splits not parallel with axes is computationally/algorithmically more challenging but has potential to produce better results. See for example [**Murthy˙1994.**]

3.4 Random Forests

Bagged version of decision trees.

3.5 Gradient Boosted Trees

Pseudocode for GBM [EH16]

1. Select tree depth D and number of iterations K
2. Compute average response $\bar{y}$ and use this as initial predicted value.
3. for $i = 1$ to K:
   (a) Compute the residual (observed - prediction), for each sample.
   (b) Fit a regression tree of depth D using residuals as the response.
   (c) Predict each sample using the regression tree fit in the previous step.
   (d) Updated predicted value of to the predicted value generated in previous step.

3.6 Bayesian Additive Regression Trees (BART)

4 Classification

4.1 Metrics

4.1.1 Area under ROC curve

Plot the True Positive Rate (aka Recall) vs False Positive Rate.

AUC is scale-invariant. It measures how well predictions are ranked, rather than their absolute values. AUC is classification-threshold-invariant. It measures the quality of the model's predictions irrespective of what classification threshold is chosen.

Classification-threshold invariance is not always desirable. In cases where there are wide disparities in the cost of false negatives vs. false positives, it may be critical to minimize one type of classification error. For example, when doing email spam detection, you likely want to prioritize minimizing false positives (even if that results in a significant increase of false negatives). AUC isn't a useful metric for this type of optimization.

4.1.2 Precicion-Recall Curve

4.1.3 Mathews Correlation Coefficient

The Pearson product moment correlation coefficient between actual and predicted values [CJ20, Methods]. As a function of the Confusion Matrix, entries, this is:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN))}}$$

4.2 Support Vector Machines (SVM)

[Bur04]

4.3 Naive Bayes Classifying

4.4 References

Burges, Christopher (2004). "A Tutorial on Support Vector Machines for Pattern Recognition". In: *Data Mining and Knowledge Discovery* 2, pp. 121–167.

Chicco, Davide and Giuseppe Jurman (Dec. 2020). "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation". In: *BMC Genomics* 21.

5 Unsupervised Learning

5.1 k-means

5.1.1 Lloyd's algorithm

- Initialize k points C.
- **repeat**
  - For all $x \in X$, find the closest center $\phi_C(x) \in C$ to x
  - Recalculate the new centroids $c_i' = \text{mean}\{x \in X : \phi_C(x) = c_i\}$.
- **until** The set C is unchanged.

There are finitely many distinct cluster centers, and the cost is always decreasing, so this is a finite algorithm. LLoyd's algorithm has complexity is $\mathbf{O}(K \cdot I \cdot N \cdot M)$ where K is the number of clusters, I is the number of iterations, N is the sample size, and M is the dimension of the space.

For initialization, there are compromises. Randomly partitioning X and taking centroids of the partitions is OK but biases towards the center of X. There is k-means++ where we choose $c_1 \in X$ randomly and then for $i = 2, \ldots, k$ we define $C_i = \{c1, \ldots, c_i\}$ and choose c_i from X with probability proportional to $d(x, \phi_{C_{i-1}}(x))^2$. So points further from any cluster are chosen more often

In order to work correctly, KMeans typically needs to have some form of normalization done of the datasets. K-means is sensitive to both means and variance in the datasets, e.g. `StandardScalar`.

Mini batch KMeans is an alternative to the traditional KMeans, that provides better performance for training on larger datasets. It is exactly what it sounds like – using small batches to update the clusters mean.

5.2 Other types of clustering

- hierarchical clustering
- density-based (e.g., MeanShift, DBSCAN)
- distribution based (e.g. GMM)
- Affinity propagation

6 Information Theory

6.1 Kullback-Liebler (KL) distance

Following [Duc16, §2.1], let X be a random variable (with countable support for now) with distribution P and pmf $P(X = x) = p(x)$. The **entropy** of X (or of P) is defined as:

$$H(X) = -\sum_x p(x) \log p(x)$$

Roughly speaking (formalized by Shannon's source-coding theorem), if we wish to encode X with distribution P with a k-ary string then the minimal expected lenght of the encoding is $H(X)$, and this is achieved by Huffman codes.

The *conditional entropy* is the amount of information left in a random variable after observing another:

$$H(X|Y = y) = -\sum_x p(x|y) \log p(x|y),$$

and

$$H(X|Y) = \sum_y p(y) H(X|Y = y)$$

For two distributions P and Q. The **Kullback-Liebler (KL) divergence** is:

$$D_{kl}(P\|Q) = \int_x p(x) \log \frac{p(x)}{q(x)}$$

Since log is concave, Jensen's inequality[2] implies that KL-divergence is non-negative and equal to 0 onlu if $P = Q$.

For finite probability space with cardinality m, it is easy to show that $D_{kl}(P\|Q) = -H(X) + \log m$.

[2]Jensen's inequality states that $f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]$ for concave f, with strict concavity implying strict inequality unless X is constant.

7 Feature Engineering

7.1 Principal Component Analysis (PCA)

8 Natural Language Processing

9 Embeddings

9.1 TF-IDF

10 Time series & Forecasting

10.1 ARIMA

An ARIMA(p, d, q) is an *autoregressive integrated moving-average* with p autoregressive terms (AR), d differencings, and q moving average (MA) terms. [HA18].

$$\phi(B)(1-B)^d Y_t = c + \theta(B)\epsilon_t$$

where

- B is the back-shift/lag operator $BY_t = Y_{t-1}$.
- $\phi(B) = (1 - \phi_1 B - \ldots - \phi_p B^p)$ is the autoregressive AR(p) component
- c is a constant
- $\theta(B) = 1 + \theta_1 B + \ldots \theta_q B^q$ is the moving average of the errors MA(q) component.
- ϵ_t is the error of the AR(p) model at time t
- The $(1-B)^d$ term induces d differencing

10.2 In `R`

`auto.arima` utilizes AIC and MLE to decide on best ARIMA parameters

10.3 In `Python`

10.4 References (Time Series & Forecasting)

Hyndman, R.J. and G. Athanasopoulos (2018). *Forecasting: principles and practice*. OTexts. URL: https://books.google.com/books?id=_bBhDwAAQBAJ.