Back to Top

CS146: Modern Computational Statistics

Table of Contents

Back to Top

Back to Top

Back to Top

Back to Top

Back to Top

Back to Top

Back to Top

CS146 Template

Pre-class

Study guide

Readings

Class

Back to Top

CS146 15.1 - Course Synthesis

Pre-class

Reflect on the course material and make notes on what the purpose of Computational Statistics is — in class, you will be asked to comment on this.
You will also have to explain a statistical method to someone who does not yet understand it, by drawing on the knowledge that you acquired in this course.

Study guide

The purpose of this session is to consider and discuss the question, “What is the point and purpose of computational statistics?” Use the Ghahramani video as an overview of what you have learned and consider the question above.

Readings

Ghahramani (2016) Bayesian Inference Part I, Machine Learning Summer School 2015

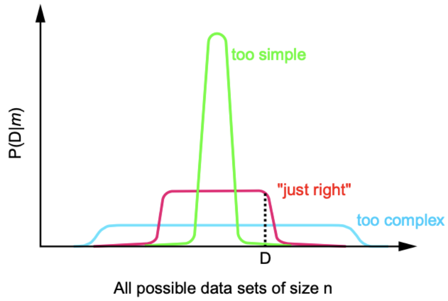
This video provides a good review of everything we've done on statistical data modeling, and also discusses the overlap between statistics and machine learning. Use the video to reflect on what we have learned about statistical data modeling in this course.
The breakdown of the video is below — please watch up to 1:19:00. The last bit of the video is optional and starts diving into the details of Bayesian inference. You can also keep watching the rest of the video series if you would like. This video is part 1 of 3.
In the latter part of the video the slides become more and more difficult to read (due to bad lighting). You can download the slides here: http://mlss.tuebingen.mpg.de/2015/slides/ghahramani/lect1bayes.pdf

- 0:00:00 Introduction
    - Views on Machine Learning
        - **Toolbox**: Toolbox of methods for processing data
        - **Modeling**: Science of learning models from data
- 0:08:00 Typical types of problems in data modeling
- 0:12:00 Bayesian inference
    - Bayes’ Rule
        - $P(hypothesis \mid data) = \frac{P(data \mid hypothesis)\, P(hypothesis)}{P(data)}$
    - Canonical ML problems
        - 1. Linear classification
        - 2. Polynomial regression
            - Parameters: coefficients
        - 3. Clustering with Gaussian mixtures
            - Parameters: Gaussian parameters, elliptical (mass) parameters, weight parameters

Back to Top

- 0:21:30 Motivation for using probability theory for data analysis, inference, and machine learning, with lots of great questions from the audience
    - Bayes Rule: can be a corollary from the sum product rule
        - Sum rule: $P(x) = \sum_y \quad P(x,y)$
        - Product rule: $P(x,y) = P(x)P(y \mid x)$
    - Prediction: marginalization over **all parameters**
        - $P(x \mid D, m) = \int \quad P(x \mid \theta, D, M) P(\theta \mid D, m)\, d\theta$
        - D: data, m: model
    - Model comparison:
        - $P(m|D) = \frac{P(D|m)P(m)}{P(D)}$
        - Marginal likelihood: $P(D|m) = \int \quad P(D \mid \theta, m) P(\theta \mid m)\, d\theta$
    - Optimization
        - Approximation to integration
        - Decision theory: When used to minimize loss
    - Representing beliefs (& **strengths** of beliefs)
        - Need to satisfy the rules of **probability theory**
    - **Asymptotic certainty**
        - $\lim_{n \to \infty} p(\theta|D_n) = \delta(\theta - \theta^*)$
            - D_n: data set, n data points
            - θ^*: true parameter
        - $\lim_{n \to \infty} p(\theta|D_n) = \delta(\theta - \hat{\theta})$
            - $\hat{\theta} = \min_{\theta} KL(p^*(x)\,, p(x|\theta))$
            - ⇒ in the worst case, you'll converge to the maximum likelihood
    - **Asymptotic consensus**
        - For two **different priors** $p_1(\theta), p_2(\theta)$ who observe the **same data**
        - The **posteriors** will converge to the same
- 0:44:00 Model selection/comparison with a demonstration using polynomial regression
    - Model fitting
        - Want to avoid **underfitting** & **overfitting**
    - Model selection
        - How many **clusters** in the data?
        - Intrinsic **dimensionality** of the data?
        - **Variable selection:** How much of/what data is relevant to the output prediction?
        - What **order** should the dynamical system have?
        - How many states in a HMM (Hidden Markov Model)?
        - **Blind source separation:** How many independent sources in the input?
        - What is the **structure** of the **graphical model**?
    - Model selection by Occam's Razor
        - Inherent to the **marginal likelihood** (a.k.a. **model evidence**)
            - $p(D|m) = \int \quad p(D \mid \theta, m)\, p(\theta \mid m)\, d\theta$
            - : probability of the **random parameter set** for the model will **generate D**
            - $log_2 \left(\frac{1}{P(D|m)}\right)$: **surprise** in observing data D over model m

[Back to Top](#)

        - 

- 1:03:00 Choosing priors
    - **Objective** priors
        - Have good **frequentist** properties
        - **Noninformative** priors ⇐ tries to capture **ignorance**
            - Hard to **generalize** to all parameters (which parameters are we trying to be noninformative about?)
        - **Reference** priors:
            - The prior should provide the **least** amount of information
            - ⇔ will provide the **most** amount of information from the **data**
            -
    - **Subjective** priors
        - Capture **beliefs**
    - **Hierarchical** priors
        - Multiple **levels** of priors (hyperpriors)
    - **Empirical priors**
        - Learn some of the parameters from data
- 1:19:40 (Optional) Computational problems in Bayesian inference

Class

Back to Top

CS146 14.2 - MCMC Sampling in Stan

Pre-class

https://docs.google.com/document/d/1ZQ8BU5sEydLxp2Ak2c43xx1CzEg-sJypVsMT6ylY0as/edit?usp=sharing

Study guide

- **Hamiltonian Monte Carlo (HMC)** optimization parameters:
  - 1. ε: **Integration time**: similar to **length parameter**
  - 2. $\mathbf{L}$: **Number of steps** between samples
  - 3. Σ: **mass matrix**: similar to the **covariance** matrix of the distribution being sampled
  - ⇒ Stan: optimized during the **warm-up** phase (warm-up samples are later discarded)
- Stan: estimating **correlation** between samples
  - n_{eff}: [n_eff] **effective number of samples**: how many samples are **uncorrelated / independent**
    - ⇒ needs to be at least a few hundred
  - $\hat{R}$: [rhat] estimation of **MCMC convergence**: average variance of each chain divided by the variance of all samples from all chains
    - ⇒ need to be close to 1. >1.1 is not mixing well

Readings

MacKay (2012) Lecture 13: Approximating Probability Distributions (III): Monte Carlo Methods (II): Slice Sampling, Hybrid Monte Carlo, Over-relaxation, Exact Sampling

A lot of this video is optional, so please read the breakdown below carefully.

- 0:06:45 — The video starts with a quick overview of the methods you have seen so far.
- 0:11:00 — Problems with methods you have seen so far, which provides motivation for better methods.
  - Problems with standard Monte Carlo
    - Random walk behavior ⇒ **efficient** methods
    - Sensitivity to **step size** ⇒ **slice** sampling
    - When to **stop** ⇒ **exact** sampling
- 0:12:45 — (Optional) Random walk behavior in 1 dimension
  - For step size $s_t = \pm 1$
  - Distance $\triangle x = \sum_{t=1}^{T} \quad s_t$
  - Variance $var(\triangle x) = \ E\left[\triangle x \ \ ^{2}\right] = \sum_{t=1}^{T} \quad E[{s_t}^2] = T$
  - $\therefore for\ var(\triangle x) = L^2, T = (L/\epsilon)^2$

Back to Top

- 0:33:20 — Random walk behavior in more than 1 dimension and why length scale matters, especially in higher dimensions.
    - For higher-dimensions with
        - γ dimensions with length ℓ
        - $\mathcal{k}$-γ dimensions with length $\mathbf{L}$ ($\mathbf{L}$>ℓ)
        - ⇒ need at least $T \simeq (L/\epsilon)^2$steps to get a **fresh independent** sample
        - ⇒ for **large** step size $\epsilon >> l$: the probability of acceptance $=$ $(l/\epsilon)^\gamma$
        - ⇒ usually, optimal step size: $\epsilon \simeq l$
            - ⇒ resulting optimal acceptance probability = 50%
- 0:41:10 — Using Hamiltonian Monte Carlo to make MCMC more efficient. The method introduces more variables (so-called momentum variables) to make sampling work better. You will need to know some basic physics to understand this fully, but if not focus on the demonstration and the idea that Hamiltonian MC uses Newtonian laws of motion to simulate movement on the surface of the log-probability density function. Amazingly this results in a valid MCMC method. You do not have to understand the details of Hamiltonian Monte Carlo to be ready for class, but you should have a conceptual understanding of how it improves on the Metropolis-Hastings method.
    - **Hamiltonian Monte Carlo** (a.k.a. **Hybrid** Monte Carlo)
        - 1. simulate Newton's laws
        - 2. accept/reject based on the changes in **total energy**
        - 3. Randomize the momentum
- 0:57:40 — (Optional) Overrelaxation for improving Gibbs sampling.
- 1:07:50 — (Optional) Slice sampling, which is a 1-dimensional MCMC method that is a huge improvement over rejection sampling.
- 1:22:10 — (Optional) Exact sampling, which allows us to draw samples as if we ran MCMC for an infinite amount of time. It generates samples that come from the target distribution exactly (number of steps, n = ∞) rather than approximately (n → ∞).
- 1:35:00 — (Optional) Exact sampling for Ising models, which is a demonstration of using exact sampling to solve a very difficult sampling problem in statistical physics.

Lambert (2018) The intuition behind the Hamiltonian Monte Carlo algorithm

This video explains how inspiration from physics was used to develop the Hamiltonian Monte Carlo sampling method. You do not have to understand all the mathematical details. Focus on developing a conceptual understanding of how incorporating the concept of momentum from physics helps generate better samples by reducing random walk behavior.

- Posterior distribution
    - Mirror & take log
- **Statistical mechanics**
    - **Canonical distribution**: the distribution of the energy level for each particle
    - Probability of the particle being at an energy level: $p(E_i) \propto e^{-E_i/T}$
        - ⇒ decays inversely to the energy level
        - ⇒ decays **slower** at **higher temperature**
- HMC

Back to Top

- Energy level: $E(\theta, M) = U(\theta) + K(M) = constant$
  - θ: parameter; M: momentum (both are vectors)
  - $U(\theta)$: potential energy= $-\ log\ (p(x|\theta) \cdot p(\theta))$
    - ⇒ the space within which the particle moves: **negative log posterior** (NLP) space
  - $K(M)$: kinetic energy= $\sum_{i=1}^{d} \frac{1}{2}\frac{{M_i}^2}{mass}$
    - Over *d* dimensions
  - ⇒ $p(E_i) \propto p(x|\theta) \cdot p(\theta) \cdot e^{-M^2/2}$
    - ⇐ assuming temperature $T = 1$, mass $mass = 1$, dimensions=1
    - ⇒ a **normal distribution**!!! (see the exponential form)
      - ⇒ can use the standard normal distribution
    - $\equiv p(x|\theta) \cdot p(\theta) \cdot N(M|0,1)$
      - The **marginal density** to our parameter:

        $$p(\theta) = \int p(\theta, M)dM = p(x|\theta) \cdot p(\theta) \int N(M|0,1)dM$$
        $$= (x|\theta) \cdot p(\theta)$$

        Is **equal** to the **posterior** density
    - ⇒ simply **sample** from the **joint distribution** and throw away the M-dimensions
    - ⇒ solve for the **path of the particle** (which is **deterministic**)
      - ⇐ use **discrete paths** (a.k.a. "**Leapfrog** algorithm")
    - ⇒ get proposed values (θ^*, M^*)
  - ⇒ work Metropolis algorithm
    - Take ratio r = (metropolis term) x (hastings term)
      - **Metropolis** term: $\{p(x|\theta^*) \cdot p(\theta^*) \cdot N(M^*|0,1)\}/\{p(x|\theta) \cdot p(\theta) \cdot N(M|0,1)\}$
      - **Hastings** term (symmetry): $p(\theta, M|\theta^*, M^*)/p(\theta^*, M^*|\theta, M)$
        - But because we traversed **deterministically** (θ,M) → (θ^*,M*),
          - $p(\theta^*, M^*|\theta, M) = 1$
          - $p(\theta, M|\theta^*, M^*) = 0$
    - Use $(\theta^*, -M^*)$as the substitute endpoint!
      - $p(\theta^*, -M^*|\theta, M) = 1$
      - $p(\theta, M|\theta^*, -M^*) = 1$
      - ⇒ Hastings term is now 1!
      - (also, $N(M^*|0,1) = N(-M^*|0,1)$)
    - Accept by $r > u \sim U(0,1)$

Betancourt (2018) Scalable Bayesian inference with Hamiltonian Monte Carlo

This video discusses the intuition behind Hamiltonian Monte Carlo as a more efficient way to do Markov Chain Monte Carlo than simple Metropolis-Hastings. It assumes you understand the goal behind MCMC and discusses the idea of physical simulation for MCMC. This should help you to understand the physical

Back to Top

idea behind Hamiltonian Monte Carlo, understand Hamilton's equations of motion as a proposal distribution, and connect Hamiltonian Monte Carlo to the Metropolis-Hastings algorithm.

-

Stan Development Team (2017) Stan Modeling Language: User's Guide and Reference Manual

Chapter 30 in the Stan manual explains what the n_eff (effective number of samples) and R-hat columns in Stan output mean. Read Sections 30.1 and 30.2 for an introduction. Focus on Section 30.3 to understand how R-hat is calculated and how it should be interpreted, Focus on Section 30.4 to understand how the effective number of samples is estimated — note how it is related to the autocorrelation of samples.

- $\hat{R}$[rhat]: average variance of samples **within each chain** to the variance of the **pooled samples**
    - **1** when converged and variances are the same (recommended **<1.1**)
    - **>1** when not converged
    - **Between sample** (B): N/(M-1) * Σ(per-chain average - across-chain average)^2
        - $B = \frac{N}{M-1}\sum_{m=1}^{M} \left(\underline{\theta}_m^{(\cdot)} - \underline{\theta}_{\cdot}^{(\cdot)}\right)^2$
            - M: Markov chains
            - N: number of samples
    - **Within-sample** (W): 1/M * Σ(per-chain value - per-chain average)^2
        - $W = \frac{1}{M}\sum_{m=1}^{M} {s_m}^2$
            - ${s_m}^2 = \frac{1}{N-1}\sum_{n=1}^{N} \left(\underline{\theta}_m^{(n)} - \underline{\theta}_m^{(\cdot)}\right)^2$
    - **Potential scale reduction statistic**:
        - $\hat{R} = \sqrt{\widehat{var}^{+}(\theta|y)/W}$
            - $\widehat{var}^{+}(\theta|y) = \frac{N-1}{N}W + \frac{1}{N}B$ (weighted mean of W and B)
    - **Split** $\hat{R}$for **non-stationarity**
        - ⇒ if, in two chains, one is gradually increasing while the other is gradually decreasing
    - Convergence is **global**
        - **All parameters** & generated quantities must converge
            - b/c different chains are in different parts of the space
    - Inconsistent $\hat{R}$?
        - There is always **residual effect** of the **initial state**
            - ⇒ particularly to functions susceptible to **large outliers**
        - $\hat{R}$ assumptions:
            - 1. Functions being considered are Gaussian, or,
            - 2. Functions uses only the first two moments ⇒ assume independence
- n_{eff}[n_eff]: **effective sample size** (ESS)
    - Samples are typically **autocorrelated within a chain**
        - ⇒ increase **uncertainty** of the **estimation** of posterior quantities
    - ⇒ number of **independent** samples
    - **Thinning samples**

Back to Top

        - ⇒ autocorrelation ρ_t decreases as the lag t increases
        - ⇒ increase n_eff

Chapter 34 explains how Hamiltonian Monte Carlo (also explained in the videos above) is implemented in Stan. You do not have to know the mathematical details. Focus on Section 34.2 about the parameters that have to be optimized to make Hamiltonian MC work well. You will see ε, which is similar to the length scale you have seen before, L, which is the number of steps between samples. The Σ matrix is new. How does Stan optimize these parameters automatically? You will be asked about this in a class poll.

-

Lambert (2018) Bob's bees: The importance of using multiple bees (chains) to judge MCMC convergence

Use this video to get an intuitive understanding for why multiple chains are used in MCMC sampling. This video should provide some intuition for why multiple chains are used to compute the R-hat value you read about in the Stan manual.

-

Feng (n.d.) The Markov-chain Monte Carlo Interactive Gallery.

This is the same sampling visualization we saw last time but using the No-U-Turn sampler (NUTS) instead of Metropolis-Hastings. Change the "target distribution" value to see how the NUTS algorithm performs on different types of distributions. Also, note how much faster NUTS moves between distant parts of the distribution compared to Metropolis-Hastings. This means **consecutive samples are much less correlated when using NUTS**.

You might want to turn the "Autoplay delay" setting to 0 to speed up the animation.

For the NUTS algorithm, the wiggly lines show the path that Stan takes for each sample. Each path has a number of steps and a step size. These are two of the parameters tuned by Stan during warm-up.

The "**no U-turn**" part of the sampler is that it makes a **wiggly line in both directions** away from the current sample location and attempts to estimate when the path turns around on itself (makes a U-turn). Since we want to move as far away from the current location as we can (so that the Markov chain mixes well) we want to avoid U-turns. As soon as Stan detects a U-turn it will stop the path and place the next sample at the location just before the U-turn.

-

McElreath (2017) Markov chains: Why walk when you can flow?

(Optional) This blog post demonstrates why modern Monte Carlo sampler, typically implementing a variation of Hamiltonian Monte Carlo, are so much more efficient than older Monte Carlo samplers. Focus on how using properties of the "surface" of the probability density function helps us make large steps from one side of the distribution to the other. This post uses some of the Chi Feng visualizations you have already played with.

-

Back to Top

Betancourt (2017) Robust statistical workflow with RStan.

(Optional) This article walks through fitting a non-trivial model using Stan and discusses the more advanced diagnostic tools available for checking whether or not the sampler converged properly. You should focus on Section 2, especially the parts on validating the model and using diagnostics. This material is not required for completing this course – you can use the simpler diagnostics we discuss in class. However, it might help you debug your model in your final project more successfully.

-

Class

Back to Top

CS146 14.1 - Metropolis-Hastings Algorithm

Pre-class

https://gist.github.com/AntonioStark/b9199be56068bb77a31b1157b05088d3

Study guide

- **Markov chain Monte Carlo (MCMC)**
    - Monte Carlo: random numbers to generate samples
    - Markov Chain: next sample can depend (only) on the **previous** sample
        - ⇒ **non-independence**
    - **"Mixing well"**: if MCMC produces **uncorrelated/independent** samples
        - **Not** mixing well: samples are correlated or does not (yet) generate samples from the target distribution
        - **Is** mixing well**:** samples are uncorrelated and from the target distribution
        - Usually takes time to start mixing well
    - Works well in
        - **High-dimensional** problems
        - Lot of **freedom** in choosing the **proposal distribution**
- **Proposal distribution**
    - Defined as ***q***
        - Defined over the next state x
        - Defined over the previous state x_0
        - PDF function: *$q(x,x_0)$*
            - Better functions ⇒ less iterations for samples to become independent
- **Metropolis-Hastings algorithm**
    - Pick initial state x_0
    - For t=0,1,...
        - Draw $x^* \sim q(x, x_0)$
        - Calculate **acceptance rate** $a = p(x^*)/p(x_t) \cdot q(x_t, x^*)/q(x^*, x_t)$
            - If **proposal** distribution is **symmetric**, second term is 1
            - ⇒ simply the "Metropolis method"
        - Accept new sample with probability $min(1, a)$
            - If accepted: $x_{t+1} = x^*$
            - If rejected: $x_{t+1} = x_t$

Back to Top

Readings

Betancourt (2018) Scalable Bayesian inference with Hamiltonian Monte Carlo

After a short introduction, the first 18 minutes of this video explains why integration (computing expectations) in high dimensions is hard and how Markov chain Monte Carlo methods can help. The rest of the video is on Hamiltonian Monte Carlo, which is the topic of the next session. You should watch the first 18 minutes of the video today and can continue with the rest in the next session.

- Bayesian inference = use **posterior** that infers from **data**
    - **Posterior** by itself is **meaningless**/useless
        - ⇒ need **expectation values** $E_\pi[f] = \int \quad dq\, \pi(q|D)\, f(q)$
    - Difficult b/c of **high-dimensional integration**
        - ⇐ no **analytical** solution
        - High**er** dimensions: small**er** **relative importance** of **volume**
        - **Dominant factors** of the **integral**
            - ⇒ not by **probability density**
            - ⇒ but by **probability mass** dq
                - ⇒ is an hypersurface: **typical set**
                    - Gets **thinner** with **higher** dimensions
- Monte Carlo simulations
    - Average a given **function** over **samples** to approximate expectations
        - $\frac{1}{N}\sum_{n=1}^{N} \quad f(q_n) \sim N\left(E[f], \frac{var[f]}{N}\right)$
        - ⇐ the expectation is **unbiased**
        - ⇐ the **error** can be **quantified**
    - ⇒ but **cannot draw** exact **samples**
- **Markov chain**
    - Can draw **correlated samples**
    - **Transition operator** that always preserves the **target distribution**
        - Is **deterministic**
        - When inside the **typical set**, would not leave
    - ⇒ if **run long enough**, the Markov Chain is consistent
        - Only an **asymptotic guarantee**
        - $\lim_{N\to\infty} \frac{1}{N}\sum_{n=0}^{N} \quad f(q_n) \ \to E_\pi[f]$

Lambert (2018) An introduction to the random walk Metropolis algorithm

An introduction to the Metropolis-Hastings algorithm. Pay attention to how the next state depends on the previous state and, in particular, how the algorithm moves to the next state with a particular acceptance probability, called r in this video. Knowing how the acceptance probability is computed is important for today's session.

- Use only the **numerator**
- **MCMC: 1st-order** Metropolis algorithm

Back to Top

        - Dependent **only on** the **first previous** sample
- Algorithm
    - Sample $\theta_0 \sim \pi(\theta)$
        - ⇐ great if pi is close to the posterior, but we usually don’t know what the posterior looks like, so we choose random
    - For iteration t, $\theta_t{}' \sim N(\theta_{t-1}, \sigma)$
        - ⇐ commonly use **normal distribution** for its **symmetricity**
            - I.e. $p(\theta_a, \theta_b) = p(\theta_b, \theta_a)$
    - Calculate ratio
        - $r = \{p(x|\theta_t') \cdot p(\theta_t')\}/\{p(x|\theta_{t-1}) \cdot p(\theta_{t-1})\}$
            - Posterior_current * prior_current / posterior_previous * prior_previous
    - Determine **accept/reject**
        - If **acceptance ratio** $r > uniform(0,1)$
            - $\theta_t = \theta_t'$
        - Else,
            - $\theta_t = \theta_{t-1}$
            - I.e. sample from same place twice

Lambert (2018) The importance of step size for random walk Metropolis

In order to be prepared for class, you need to understand the importance of the step size parameter of the Metropolis-Hastings method and the problems associated with picking a step size that is either too large or too small. The importance of an appropriate step size is also the main topic of the pre-class work.

- Metropolis algorithm
    - **Jumping kernel** requires **symmetry**
        - I.e. $J(\theta_t|\theta_{t-1}) = J(\theta_{t-1}|\theta_t)$
        - ⇒ usually use **normal distribution**
            - $\theta_t' \sim N(\theta_{t-1}, \sigma)$
    - Guaranteed to **asymptotically converge**
    - For **finite sample size**, performance is measured by **convergence speed**
        - **Small** step size δ: accept higher proportions of steps, but take longer time to explore posterior space
        - **Large** step size δ: accept low proportions of steps, but take shorter time to explore posterior space

Feng (n.d.) The Markov-chain Monte Carlo Interactive Gallery

This demonstration shows how the standard Metropolis-Hastings algorithm performs on different types of distributions. Change the "target distribution" value to see how the Metropolis-Hastings algorithm performs on different types of distributions. Understand what is meant by the random walk behavior of Metropolis-Hastings, especially for a small step size, called the "proposal σ" here. Change the proposal σ and predict the parameter's effect on the algorithm.

Back to Top

Watch 0:53:20–1:08:20 MacKay (2012) Lecture 12: Approximating Probability Distributions (II): Monte Carlo Methods (I): Importance Sampling, Rejection Sampling, Gibbs Sampling, Metropolis Method

This video describes and demonstrates the Metropolis-Hastings algorithm. This video accompanies the MacKay reading below.

- Acceptable for the proposal distribution *q* to be
    - Simple or complicated
    - **Not look like** the target distribution *p*
- Each t:
    - Sample $x \sim q(x; x^t)$
    - Calculate **acceptance ratio**
        - Importance ratio $p^*(x)/q(x \quad ; x^t)$
        - Weight (how likely is it for us to back-sample to the original?) $q(x^t; x)/p^*(x^t)$
        - ⇒ no need to know the **normalization constant** of the **proposal** distribution q
            - (as long as the normalization constant **does not depend** on x_t)
        - **Without** acceptance ratio, it would be **complete random walks**
            - ⇐ **no** target distribution (p) involvement
            - **<u>If q is symmetric</u>**, the $q(x^t; x)/q(x; x^t) = 1$
    - Determine accept/reject
        - Always accept if $a > 1$
        - If not, accept with probability a
            - Reject: $x^{t+1} = x^t$ (same place sampling)

Section 29.4 Mackay (2003) Information theory, inference, and learning algorithms

After having gone through the introductory readings above, this section is the main reference for the Metropolis-Hastings method. You should read this section very carefully and do the exercises to improve your understanding of how the algorithm proceeds and why the length scale of the proposal distribution is important to the convergence of the algorithm.

- Importance sampling / rejection sampling
    - Work well only if proposal q(x) is similar to target p(x)
- Rule of thumb: the **lower bound** on the number of iterations for Metropolis-Hastings
    - $T \simeq (L/\epsilon)^2$
        - T: time
        - L: the largest length scale of the space
        - ε: random walk step size
    - ⇒ the time required to get a sample roughly independent of the initial condition
    - ⇐ assumes every step is accepted
        - If fraction *f* of the steps are accepted, on average,
        - Time is increased by factor 1/*f*

Back to Top

(Optional) Wiecki, T. (2015, November 10). MCMC sampling for dummies.

This article explains Metropolis-Hastings and implements the algorithm in Python. Use it to understand how Metropolis-Hastings is implemented and how to summarize the results from MCMC by plotting your samples.

-

Class

Back to Top

CS146 13.2 - Convergence of Importance Sampling

Pre-class

https://docs.google.com/document/d/1i89ZBiqUpPsdOcQGJPEF4nr7dqmg7-Cnbbh0tlsAEDg/edit?usp=sharing

Study guide

- Impact of **proposal distribution choice** to convergence results
- **Importance sampling**
    - Generate n samples$\{x_i\}$from the proposal distribution
    - Calculate weights w_i per sample
    - Compute weighted average $E[f(x)] \approx [\sum_{i=1}^{n} \quad w_i \cdot f(x_i)]/[\sum_{i=1}^{n} \quad w_i]$
- Approximation of **nth moments** available by setting $f(x) = x^n$
    - $variance \; = \; (second\ moment) - (first\ moment)^2$
- **Normalization constant** for $\bar{p}(x)$
    - $C = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \quad w_i$
        - $w_i = \underline{p}(x_i)/q(x_i)$
        - Target distribution (unnormalized): $\underline{p}(x) = K \cdot p(x)$
            - Target distribution (normalized): $p(x)$
        - Proposal distribution (normalized): $q(x)$
- Convergence
    - Identify via **partial sums** of the approximations
    - Absolute error of an estimate
        - Scale by $n^{-1/2}$

Readings

MacKay (2012) Lecture 12: Approximating Probability Distributions (II): Monte Carlo Methods (I): Importance Sampling, Rejection Sampling, Gibbs Sampling, Metropolis Method

The video associated with MacKay Chapter 29. The video covers a number of topics in Monte Carlo methods, but for now focus on the segment 0:30:00–0:54:00, which covers the problems we are trying to solve with sampling, importance sampling, and rejection sampling.

- Types
    - **Simple** Monte Carlo: importance sampling; Rejection sampling
    - **Markov-chain** Monte Carlo: Metropolis method, Gibbs sampling, slice sampling
    - **Random-walk** behavior reduction: Hamiltonian MC, Overrelaxation
- **Proposal** distribution (Q): **sampler density**
    - ⇒ `can` **draw samples**

Back to Top

    - Converges to $\frac{1}{R}\sum_{i=1}^{R} \quad \phi(x^{(r)})$ ⇒ but is the wrong convergence point
    - $\left[\sum_{i=1}^{R} \quad \frac{p^*(x^{(r)})}{q(x^{(r)})}\phi(x^{(r)})\right] / \left[\sum_{i=1}^{R} \quad \frac{p(^*x^{(r)})}{q(x^{(r)})}\right]$
        - $E\left[\sum_{i=1}^{R} \quad \frac{p^*(x^{(r)})}{q(x^{(r)})}\phi(x^{(r)})\right] \rightarrow \int_x \quad P^*(x)/Q(x)\phi(x) \cdot Q(x)dx = \int_x \quad P^*(x)\phi(x)dx$
- Monte Carlo
    - Will **asymptotically converge**
        - Sometimes will **"lurch"** around when it finally hits on the very rare sample with P*>>Q (i.e. very large weights)
    - But **no guarantee** on whether it will converge to the **correct value**
- Rejection sampling
    - Target distribution $P^*(x)$
    - Proposal distribution $Q(x)$
        - $\exists c \; \forall x \; cQ(x) > P^*(x)$
        - ⇒ cannot do rejection sampling when Q(x) does not exist
        - ⇒ or when c is too big (i.e. rejects almost every sample)
    - Process
        - Draw $x_i \sim Q$
        - Draw $u_i \sim Uniform(0, cQ(x_i))$
        - Accept x_i if $u_i < P^*(x_i)$

[mathematicalmonk] (2011) Importance sampling — introduction

This is a series of three videos on importance sampling. Watch all three videos to get an introduction to the purpose, inner workings, and problems associated with importance sampling. For today the focus is on the convergence of importance sampling and how the choice of proposal distribution can affect this convergence. A bad proposal might mean that we converge to the wrong value, a good proposal could converge faster even than sampling from the true target distribution.

- Monte Carlo methods
    - $E[f(x)] \approx \frac{1}{n}\sum_{i=1}^{n} \quad f(x_i)$
        - ⇐ assumes $x \sim p, x_i \sim p$
- Importance sampling
    - **Not** a sampling method
        - A **Monte Carlo approximation** method
    - Approximation process
        - $E[f(x)] = \int_x \quad f(x) \cdot p(x)\, dx$
            - $p(x)$ being a **probability distribution**
        - $= \int_x \quad f(x) \cdot p(x)/q(x) \cdot q(x)\, dx$
            - $q(x)$being a **probability distribution** (and a **proposal distribution**)
            - **Absolute continuity**: $\forall x \; s.t. q(x) = 0 \Rightarrow p(x) = 0$
        - $= \int_x \quad F(x) \cdot q(x)\, dx$
            - $F(x) = f(x) \cdot p(x)/q(x)$
        - $= E[F(x)]$
            - $\because$ $q(x)$is also a **valid probability distribution**
        - $\approx \frac{1}{n}\sum_{i=1}^{n} \quad F(x_i^q)$

Back to Top

-
  -
    - $=\frac{1}{n}\sum_{i=1}^{n} \quad f(x_i^q)\cdot w(x_i^q)$
      - x_i is now sampled from our **proposal distribution** (q)
      - **Importance weights** $w(x) = p(x)/q(x)$
  - **Unbiased**
    - expectation value of the estimator is the same as the expectation value of the original
      - Estimator: $I = \frac{1}{n}\sum_{i=1}^{n} \quad f(x_i^q)\cdot w(x_i^q)$
      - Original: $E[f(x)] = \int_x \quad f(x)p(x)dx$
  - **Variance**
    - Variance of the estimator
      - $\sigma^2(I) = \frac{1}{n}\left(f(x)\frac{p(x)}{q(x)}\right)$
      - ⇒ can be **smaller** than the original variance
    - Variance of the original
      - $\sigma^2(\hat{\mu}_n) = \frac{1}{n}\sigma^2(f(x))$
- Choosing **proposal distribution** for **importance sampling**
  - Choose where $|f(x)|p(x)$ is large
  - ⇒ this will **minimize variance**

(Optional) Chapter 29 up to the end of Section 29.3, MacKay (2003) Information theory, inference, and learning algorithms

This reading and the video below go together. Briefly review the sections on importance and rejection sampling. Focus on the factors that influence the convergence of these methods. We discuss convergence of Monte Carlo methods in more detail in class.

- Introduction
  - When **Gaussian** approximations are **inaccurate**
  - Sample: denotes **single** datum x whose probability distribution is $P(x)$
  - **Transition probability**
    - Transition **probability matrix**: $T_{ij} = T_{i\,|\,j}$ for $j \rightarrow i$ (current state is j)
    - Transition **probability density**: $T(x'; x)$ for $x \rightarrow x'$
- Monte Carlo problem space
  - For **generating samples** $\{x^{(r)}\}_{r=1}^{R}$
    - Given the **target density** $P(x)$
  - For **estimating expectations** $\Phi = \langle\phi(x)\rangle \equiv \int \quad d^N x\, P(x)\, \phi(x)$
    - If samples $\{x^{(r)}\}_{r=1}^{R}$ can be generated from $P(x)$,
      - **Estimator** $\hat{\Phi} \equiv \frac{1}{R}\sum_r \quad \phi\left(x^{(r)}\right)$
        - $E[\Phi] = \hat{\Phi}$
      - Variance $var(\hat{\Phi}) = \sigma^2/R$
        - $\sigma^2 = var(\phi) = \int \quad d^N x P(x)(\phi(x) - \Phi)^2$
- **Importance sampling**
  - Only about **estimating the expectation** of $\phi(x)$ (not about generating samples from $P(x)$)
  - Assumptions:

Back to Top

        - We can evaluate the **density** of the function at **any point** to a **multiplicative constant** (i.e. un-normalized)
            - $\exists P^*(x)\ such\ that\ P(x) = P^*(x)\ /\ Z$
        - We can **generate samples** from a **simpler** density function, **sampler density**
            - $\exists Q(x)$
        - We can **evaluate** to a multiplicative constant for the **sampler density**
            - $\exists Q^*(x)\ such\ that\ Q(x) = Q^*(x)\ /\ Z_Q$
    - Generating samples
        - Generate R samples $\{x^{(r)}\}_{r=1}^{R}$ from $Q(x)$
        - These samples **over-represent** $x, Q(x) > P(x)$
        - These samples **under-represent** $x, Q(x) < P(x)$
        - **Weights** $w_r \equiv [P^*(x^{(r)})]/[Q^*(x^{(r)})]$
        - ⇒ adjust the importance of estimators $\hat{\Phi} \equiv \sum_r \quad \phi(x^{(r)}) = (\sum_r \quad w_r\phi(x^{(r)}))/(\sum_r \quad w_r)$
    - Reliability of estimator $\hat{\Phi}$
        - If **proposal density** $Q(x)$ is **small**
            - for where $|\phi(x)P^*(x)|$ is large,
            - After many $x^{(r)}$generated samples, none may have landed in the **typical set**
        - ⇒ importance sampler should have **heavy tails** (e.g. Cauchy over Gaussian)
- Importance sampling in **multidimensions**
    - Obtaining samples that lie in the **typical set of P** takes a long time
    - The **weights** of the probabilities in the **typical set** will differ by **factors of order** $exp\ (\sqrt{N})$
    - (unless Q is an extraordinarily good approximation to P)
- **Rejection sampling**
    - $Q(x)$: **proposal density**: can evaluate (within a multiplicative factor Z_Q) and can generate samples
        - Constant $\forall c, cQ^*(x) > P^*(x)$
    - $P(x)$: **target density**
    - Process
        - Generate sample $x \sim Q(x)$
        - Evaluate $cQ^*(x)$
        - Generate uniformly distributed random variable $u \sim [0, cQ^*(x)]$
        - ⇒ generating a sample from cQ*'s AUC
        - Evaluate $P^*(x)$
        - Accept if $u < P^*(x)$; reject if $u > P^*(x)$
- Rejection sampling in **multidimensions**
    - Constant required for $\forall c, cQ^*(x) > P^*(x)$is too large ⇒ **acceptance is rare**
        - Constant c grows **exponentially** to **dimensionality**

Class

Back to Top

CS146 13.1 - Rejection and Importance Sampling

Pre-class

https://gist.github.com/AntonioStark/1b46dc14460b472cdfa42945ce9b75c6

Study guide

- Monte Carlo sampling
    - ⇒ approximate inference when **analytical** solutions are intractable
    - (used in Stan)
- Monte Carlo sampling purposes
    - To draw **samples from a distribution** $p(x)$
    - To compute **expected values** $E[f(x)] = \int_x f(x)\, p(x) dx$
- **Rejection sampling**
    - **Proposal distribution**: usually standard distributions (b/c easy to sample from)
    - **Rejection**: if sample is above target distribution's PDF
    - ⇒ approximate the "area under the curve (AUC)"
- **Importance sampling**

Readings

Chapter 29 up to the end of Section 29.3, MacKay (2003) *Information theory, inference, and learning algorithms*

MacKay outlines why generating samples or computing expected values in complicated, high-dimensional distributions is a difficult problem, in general. There is also a short summary of rejection and importance sampling. Use this reading to get an overview of today's session. Use the study guide and the videos above for more detail. The subsection in 29.1 called "Uniform sampling" refers to material in the book that we have not covered, so you can skip it.

- Introduction
    - When **Gaussian** approximations are **inaccurate**
    - Sample: denotes **single** datum x whose probability distribution is $P(x)$
    - **Transition probability**
        - Transition **probability matrix**: $T_{ij} = T_{i\mid j}$ for $j \to i$ (current state is j)
        - Transition **probability density**: $T(x'; x)$ for $x \to x'$
- Monte Carlo problem space
    - For **generating samples** $\{x^{(r)}\}_{r=1}^{R}$
        - Given the **target density** $P(x)$
    - For **estimating expectations** $\Phi = \langle\phi(x)\rangle \equiv \int d^N x\, P(x)\, \phi(x)$
        - If samples $\{x^{(r)}\}_{r=1}^{R}$ can be generated from $P(x)$,
            - **Estimator** $\hat{\Phi} \equiv \frac{1}{R}\sum_r \phi(x^{(r)})$
                - $E[\Phi] = \hat{\Phi}$

Back to Top

            - Variance $var(\hat{\Phi}) = \sigma^2/R$
                - $\sigma^2 = var(\phi) = \int \quad d^N x P(x)(\phi(x) - \Phi)^2$
- **Importance sampling**
    - Only about **estimating the expectation** of $\phi(x)$ (not about generating samples from $P(x)$)
    - Assumptions:
        - We can evaluate the **density** of the function at **any point** to a **multiplicative constant** (i.e. un-normalized)
            - $\exists P^*(x)\ such\ that\ P(x) = P^*(x)\ /\ Z$
        - We can **generate samples** from a **simpler** density function, **sampler density**
            - $\exists Q(x)$
        - We can **evaluate** to a multiplicative constant for the **sampler density**
            - $\exists Q^*(x)\ such\ that\ Q(x) = Q^*(x)\ /\ Z_Q$
    - Generating samples
        - Generate R samples $\{x^{(r)}\}_{r=1}^{R}$ from $Q(x)$
        - These samples **over-represent** $x, Q(x) > P(x)$
        - These samples **under-represent** $x, Q(x) < P(x)$
        - **Weights** $w_r \equiv [P^*(x^{(r)})]/[Q^*(x^{(r)})]$
        - ⇒ adjust the importance of estimators $\hat{\Phi} \equiv \sum_r \quad \phi(x^{(r)}) = (\sum_r \quad w_r\phi(x^{(r)}))/(\sum_r \quad w_r)$
    - Reliability of estimator $\hat{\Phi}$
        - If **proposal density** $Q(x)$ is **small**
            - for where $|\phi(x)P^*(x)|$ is large,
            - After many $x^{(r)}$generated samples, none may have landed in the **typical set**
        - ⇒ importance sampler should have **heavy tails** (e.g. Cauchy over Gaussian)
- Importance sampling in **multidimensions**
    - Obtaining samples that lie in the **typical set of P** takes a long time
    - The **weights** of the probabilities in the **typical set** will differ by **factors of order** $exp\ (\sqrt{N})$
    - (unless Q is an extraordinarily good approximation to P)
- **Rejection sampling**
    - $Q(x)$: **proposal density**: can evaluate (within a multiplicative factor Z_Q) and can generate samples
        - Constant $\forall c, cQ^*(x) > P^*(x)$
    - $P(x)$: **target density**
    - Process
        - Generate sample $x \sim Q(x)$
        - Evaluate $cQ^*(x)$
        - Generate uniformly distributed random variable $u \sim [0, cQ^*(x)]$
        - ⇒ generating a sample from cQ*'s AUC
        - Evaluate $P^*(x)$
        - Accept if $u < P^*(x)$; reject if $u > P^*(x)$
- Rejection sampling in **multidimensions**
    - Constant required for $\forall c, cQ^*(x) > P^*(x)$is too large ⇒ **acceptance is rare**
        - Constant c grows **exponentially** to **dimensionality**

Back to Top

Lambert (2018) *An introduction to rejection sampling*

This video provides an introduction to rejection sampling. Watch it for background information and a demonstration of how this method works.

- **PDFs**: tell us how the sample distribution should **look like**, but does not tell you **how to generate** those samples
- Rejection sampling
    - For $x_i \sim distr(domain), y_i \sim distr(domain)$
    - If $y_i < p(x_i)$, accept the sample. Reject if not
- Can be **inefficient**
    - Usually rejects more data points than it **accepts**
    - ⇒ inefficiency **compounds** with **dimensionality**

Lambert (2018) *An introduction to importance sampling*

This video and the next one provide an introduction to importance sampling. Watch it for background information and a demonstration of how this method works.

- Expectation value
    - $E_f[x] = \sum_x \quad x \cdot f(x)$
    - $E_f[x] \approx \frac{1}{n}\sum_{i=1}^{n} \quad {x^f}_i$ (n is # of samples)
- Importance sampling
    - $E_g[x] = \sum_x \quad x \cdot g(x) = \sum_x \quad x\frac{g(x)}{f(x)} \cdot f(x) = E_f\left[x \cdot \frac{g(x)}{f(x)}\right] \approx \frac{1}{n}\sum_{i=1}^{n} \quad x_i^f \quad \frac{g(x_i^f)}{f(x_i^f)}$
        - x_i^f are sampled from the distribution described by $f(x)$
    - ⇒ if one knows the **ratios** between two distributions, one can use the simpler distribution to calculate the expectation for the target distribution
    - Weight $w_i = g(x_i)/f(x_i)$
- Importance sampling benefits
    - Works well for **continuous multivariate** examples (where quantity of interest may be analytically intractable)
    - $g(x)$ can be **unnormalized**
        - For $g(x) = h(x)/z$ (h being the normalized distribution, z being the normalization constant)
        - $E_g[x] \approx \frac{1}{z}\frac{1}{n}\sum_{i=1}^{n} \quad x_i \cdot \frac{h(x_i)}{f(x_i)}$
        - Can calculate the **normalization constant** by:
            - $E_g[c] = \sum_x \quad \frac{h(x)}{z} = c$ (c is a **constant** value)
            - $z = \frac{1}{c}\sum_x \quad h(x) = \sum_x \quad \frac{h(x)/c}{f(x)} f(x) \approx \frac{1}{n}\sum_{i=1}^{n} \quad \frac{h(x_i)/c}{f(x_i)} = \frac{1}{n}\sum_{i=1}^{n} \quad \frac{g(x_i)}{f(x_i)} = \frac{1}{n}\sum_{i=1}^{n} \quad w_i = \underline{w}$
        - $\therefore E_g[x] = (\, \underline{xw}\,)/\, \underline{w}$

Lambert (2018) *An introduction to importance sampling — optimal importance distributions*

This video and the previous one provide an introduction to importance sampling. Watch it for background information and a demonstration of how this method works.

- Importance sampling (discrete)
    - $E_g[x] \approx \frac{1}{n}\sum_{i=1}^{n} \quad w_i^f \cdot x_i^f$

[Back to Top](#)

        - $w^f{}_i = g(x_i^f)\ /\ f(x_i^f)$
- Importance sampling (continuous)
    - $E_g[x] = \int x \cdot g(x)\,dx = \int x \cdot \frac{g(x)}{f(x)} \cdot f(x)\,dx \approx \frac{1}{n}\sum_{i=1}^{n} w_i^f\ x_i^f$
- Importance sampling (function of variable)
    - $E_g[\gamma(x)] = \int \gamma(x) \cdot g(x)dx \approx \frac{1}{n}\sum_{i=1}^{n} w_i^f \cdot \gamma\left(x_i^f\right)$
    - ⇒ what's the w_i^f here?

Class

Back to Top

CS146 12.2 - Case study: The TrueSkill Model

Pre-class

Read through the TrueSkill paper carefully and use the study guide to help you understand how the model works and how inference is done in the model. You should be ready to discuss and critique the model in class and to understand code that implements the model. The main point of this session is for you to work through an academic paper on the topic of Bayesian inference. Papers are typically very dense and it takes work to understand one of them.

Study guide

Readings

Bishop, C.M. (2013). Graphical Models 3

Watch the segment 0:49:40–1:09:30 for an introduction to the Expectation Propagation algorithm, as applied to the TrueSkill model. We discussed Expectation Propagation in the previous session and you will work through the original TrueSkill paper in this session.
The last segment of the video, from 1:09:30 to the end, is also recommended. It connects approximate inference in graphical models to the currently active research on Probabilistic Programming, which is about automating inference as much as possible.

- TrueSkill: Xbox 360 & Xbox Live
  - For leaderboard
  - For matching players real-time
- **Elo**
  - Single rating for each player
  - International standard for chess grading
  - Not to team games or for 2+ players
- Stages of Model-based machine learning (MBML)
  - 1. **Build** the model: expressed as the **joint probability** of all relevant variables
  - 2. Incorporate **observed data**
  - 3. **Inference**: compute distributions over the variables of interest
- MBML for game-playing
  - Ranking (over all game) → performance (per game) → outcome (by whose performance is better)
  - → update distributions
- TrueSkill: converge **must faster** than Elo
  - Make very big changes when necessary

Back to Top

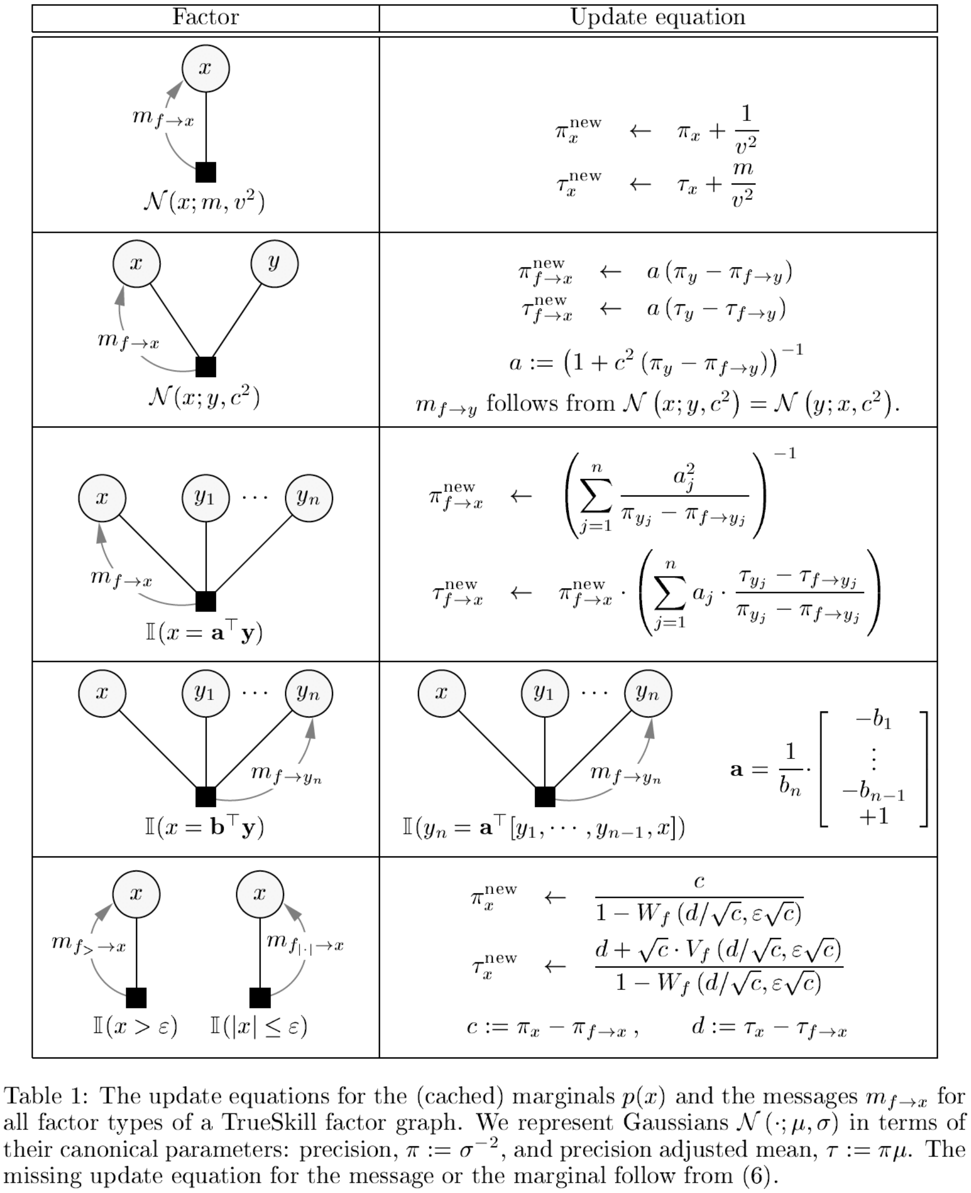
Herbrich, Minka, Graepel (2006) TrueSkill™: A Bayesian Skill Rating System

This is the TrueSkill paper that we use as a case study of applying the Expectation Propagation algorithm to a real-world problem. Focus on understanding 1) how the factor graph represents the real-world scenario of modeling the skills of players of a game, and how those skills determine the probability of one player or the other winning a game; 2) on how inference is done in the model by approximating messages in the sum-product algorithm using Normal distributions.

- Introduction: Elo
- Factor graphs for ranking
    - **Skill** (s_i)
        - Typically improves over time
            - ⇒ Gaussian dynamics factor $N(s_{i,t+1};\ s_{i,t}, \gamma^2)$
        - Incorporates time (hidden in factor graph)
        - Has a **normal prior**
            - $p(s) := \prod_{i=1}^{n} \quad N(s_i\ ;\ u_i, {\sigma_i}^2)$
    - **Performance** (p_i)
        - Normal distribution with mean $\mathbf{s_i}$
            - $p_i \sim N(p_i;\ s_i, \beta^2)$
    - **Team performance** (t_i)
        - Sum of performances of players in the team
            - $t_j := \sum_{i \in A_j} \quad p_i$
        - Probability of game outcome
            - $p(s \mid r, A) \ = \frac{P(r|s,A)\, p(s)}{p(r|A)}$
                - r: game outcome
                - s: skills of participating players
                - $A := \{A_1, \dots, A_k\}$: team assignment
    - Game outcome (r)
        - Team ranking: ranking of team performances
            - $P(r \mid \{t_1, \dots, t_k\}) = P(t_{r(1)} > t_{r(2)} > \dots > t_{r(k)})$
        - For **similar-performing teams**, outcome is a draw
            - When $|t_i - t_j| < \epsilon$
- Approximate message passing
    - Semantics
        - $f;\ f(v)$: factor node
            - F_{v_k}: set of factors connected to v_k
        - $m_{v_i \to f}$: incoming message (variable to factor)
        - v_i: variable node
            - $v_{\setminus j}$: all variable nodes except for node v_j
        - $p(v_k)$: marginal over variable v_k
    - Message passing
        - (3) $p(v_k) = \prod_{f \in F_{v_k}} \quad m_{f \to v_k}(v_k)$
        - (4) $m_{f \to v_{j(v_j)}} = \int \quad \cdots \int \quad f(v) \prod_{i \neq j} \quad m_{v_i \to f}(v_i)\, dv_{\setminus j}$
        - (5) $m_{v_k \to f}(v_k) = \prod_{\hat{f} \in F_{v_k} \setminus \{f\}} \quad m_{\hat{f} \to v_k}(v_k)$

Back to Top

- Update equations



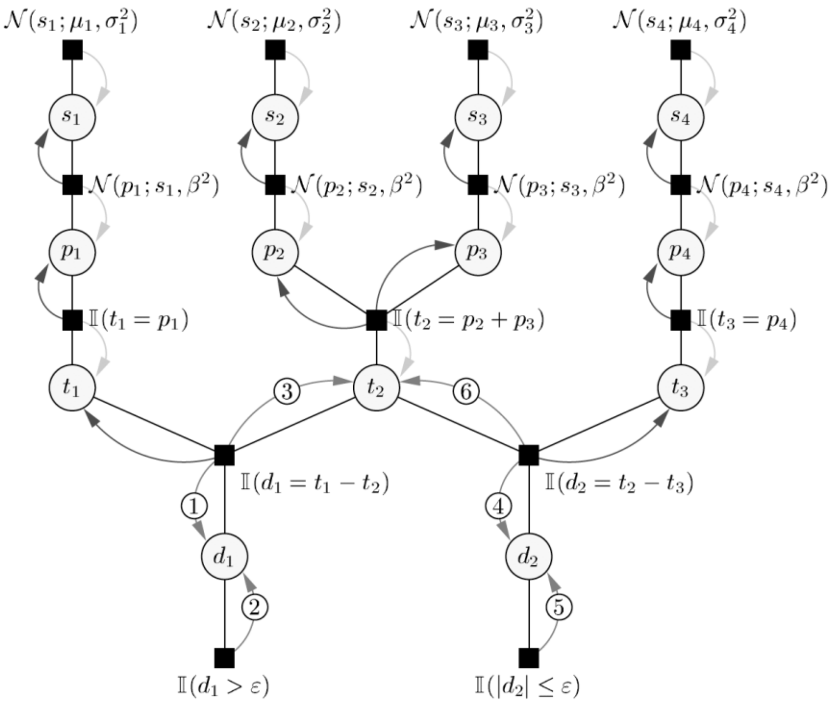
Factor	Update equation
$m_{f\to x}$ $\mathcal{N}(x;m,v^2)$	$\pi_x^{\text{new}} \leftarrow \pi_x + \frac{1}{v^2}$ $\tau_x^{\text{new}} \leftarrow \tau_x + \frac{m}{v^2}$
$m_{f\to x}$ $\mathcal{N}(x;y,c^2)$	$\pi_{f\to x}^{\text{new}} \leftarrow a\left(\pi_y - \pi_{f\to y}\right)$ $\tau_{f\to x}^{\text{new}} \leftarrow a\left(\tau_y - \tau_{f\to y}\right)$ $a := \left(1 + c^2\left(\pi_y - \pi_{f\to y}\right)\right)^{-1}$ $m_{f\to y}$ follows from $\mathcal{N}\left(x;y,c^2\right) = \mathcal{N}\left(y;x,c^2\right)$.
$m_{f\to x}$ $\mathbb{I}(x = \mathbf{a}^\top\mathbf{y})$	$\pi_{f\to x}^{\text{new}} \leftarrow \left(\sum_{j=1}^{n} \frac{a_j^2}{\pi_{y_j} - \pi_{f\to y_j}}\right)^{-1}$ $\tau_{f\to x}^{\text{new}} \leftarrow \pi_{f\to x}^{\text{new}} \cdot \left(\sum_{j=1}^{n} a_j \cdot \frac{\tau_{y_j} - \tau_{f\to y_j}}{\pi_{y_j} - \pi_{f\to y_j}}\right)$
$m_{f\to y_n}$ $\mathbb{I}(x = \mathbf{b}^\top\mathbf{y})$	$m_{f\to y_n}$ $\mathbb{I}(y_n = \mathbf{a}^\top[y_1,\cdots,y_{n-1},x])$ $\mathbf{a} = \frac{1}{b_n}\cdot\begin{bmatrix} -b_1 \\ \vdots \\ -b_{n-1} \\ +1 \end{bmatrix}$
$m_{f_>\to x}$ $m_{f_{\|\cdot\|}\to x}$ $\mathbb{I}(x > \varepsilon)$ $\mathbb{I}(\|x\| \le \varepsilon)$	$\pi_x^{\text{new}} \leftarrow \frac{c}{1 - W_f\left(d/\sqrt{c}, \varepsilon\sqrt{c}\right)}$ $\tau_x^{\text{new}} \leftarrow \frac{d + \sqrt{c}\cdot V_f\left(d/\sqrt{c}, \varepsilon\sqrt{c}\right)}{1 - W_f\left(d/\sqrt{c}, \varepsilon\sqrt{c}\right)}$ $c := \pi_x - \pi_{f\to x}\,, \quad d := \tau_x - \tau_{f\to x}$

Table 1: The update equations for the (cached) marginals $p(x)$ and the messages $m_{f\to x}$ for all factor types of a TrueSkill factor graph. We represent Gaussians $\mathcal{N}(\cdot;\mu,\sigma)$ in terms of their canonical parameters: precision, $\pi := \sigma^{-2}$, and precision adjusted mean, $\tau := \pi\mu$. The missing update equation for the message or the marginal follow from (6).

  - Precision $\pi := \sigma^{-2}$
  - Precision-adjusted-mean $\tau := \pi\mu$

- Example

[Back to Top](#)



$\mathcal{N}(s_1;\mu_1,\sigma_1^2)$
$\mathcal{N}(s_2;\mu_2,\sigma_2^2)$
$\mathcal{N}(s_3;\mu_3,\sigma_3^2)$
$\mathcal{N}(s_4;\mu_4,\sigma_4^2)$
s_1
s_2
s_3
s_4
$\mathcal{N}(p_1;s_1,\beta^2)$
$\mathcal{N}(p_2;s_2,\beta^2)$
$\mathcal{N}(p_3;s_3,\beta^2)$
$\mathcal{N}(p_4;s_4,\beta^2)$
p_1
p_2
p_3
p_4
$\mathbb{I}(t_1=p_1)$
$\mathbb{I}(t_2=p_2+p_3)$
$\mathbb{I}(t_3=p_4)$
t_1
t_2
t_3
3
6
$\mathbb{I}(d_1=t_1-t_2)$
$\mathbb{I}(d_2=t_2-t_3)$
1
4
d_1
d_2
2
5
$\mathbb{I}(d_1>\varepsilon)$
$\mathbb{I}(|d_2|\le\varepsilon)$

Figure 1: An example TrueSkill factor graph. There are four types of variables: s_i for the *skills* of all players, p_i for the *performances* of all players, t_i for the *performances* of all *teams* and d_j for the *team performance differences*. The first row of factors encode the (product) prior; the product of the remaining factors characterises the likelihood for the game outcome Team 1 > Team 2 = Team 3. The arrows indicate the optimal message passing schedule: First, all light arrow messages are updated from top to bottom. In the following, the schedule over the team performance (difference) nodes are iterated in the order of the numbers. Finally, the posterior over the skills is computed by updating all the dark arrow messages from bottom to top.

- Probability of winning
  - Indicator function: $I(p_1 > p_2)$: 1 when p1>p2 and 0 otherwise
  - Result
    - $\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty}$
      - $\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} I(p_1 > p_2)\, N(p_1 \mid s_1, \beta^2)\, N(s_1 \mid \mu_1, \sigma_1^2)\, N(p_2 \mid s_2, \beta^2)\, N(s_2 \mid \mu_2, \sigma_2^2)\, ds_1\, dp_1\, ds_2\, dp_2$

Lee, H. (2016). *TrueSkill — The video game rating system*

We run the TrueSkill model on a real sports dataset in class. You should prepare by having a brief look at the TrueSkill library described in this reading. You do not need to know everything about the library, but it will help you understand the coding activity in class if you familiarize yourself with it. You can also install the library on your laptop (pip install trueskill) and run the examples shown in this reading.

- Objects
  - Rating: skill level of player, described with mu and sigma
  -

Back to Top

Class

Back to Top

CS146 12.1 - Expectation Propagation

Pre-class

https://docs.google.com/document/d/1PjqS4vEGLvk3tUOcQTsbOKTeyq_WiAKaBwcVyi2GEt4/edit?usp=sharing

Study guide

- Expectation propagation
- Normal approximation
    - Method 1:
        - normal approximation to the **mode**
        - Match **local curvature**
        - Works well only for **unimodal** distributions
    - Method 2 (**moment matching; matching sufficient statistics**):
        - Copy mean and variance of original distribution
        - Captures **global properties** but ignores **local properties**
- **KL (Kullback-Leibler) divergence**
    - Generally, **matching sufficient statistics** is equivalent to **minimizing KL-divergence**
        - esp. For the **exponential family** of distributions
            - b/c any such distribution $\propto exp\ (\theta \cdot T(x) - f(\theta))$
                - θ: vector of natural parameters of the distribution
                - $T(x)$: vector of sufficient statistics of the data
                - $f(\theta)$: function of the parameters only
    - For probability distributions *p* and *q*,
        - $D_{KL}(p\ ||\ q)\ =\ \int_x\ \ p(x)\ log\,\frac{p(x)}{q(x)}$
- **Expectation propagation**
    - Is a **deterministic approximate** method
        - I.e. no **random sampling** involved
    - Approximates every factor in **unnormalized posterior** with a normal distribution
        - **Posterior** = product of all factors in a factor graph
    - EP: approximate on all factors except one, and compute sufficient statistics (mean and variance) of the remaining factor
        - ⇒ update approximation for remaining factor
        - ⇐ at every step, remove **one approximate factor** from the product
            - ⇒ **cavity distribution**
            - ⇒ remove one term from the mean/variance from the product of normals
        - ⇐ multiply in the **real factor** corresponding to the **approximate factor** we just removed
            - ⇒ results in a **hybrid** distribution (i.e. not a normal distribution anymore)

Back to Top

        - ⇒ `compute` **mean** and **variance** of the **hybrid distribution** and replace the real factor with a new **approximate normal** distribution to re-match the mean and variance
    - Does not **guarantee convergence** but usually works well
- Product of normal PDFs
    - $\prod_{i=1}^{n} \quad N(x \mid \mu_i, \sigma_i{}^2) \;\propto\; N(x \mid \mu_N, \sigma_N{}^2)$
        - $\mu_N = \sigma_N{}^2 \sum_{i=1}^{n} \quad (\mu_i \,/\, \sigma_i{}^2)$
        - $\sigma_N{}^2 = (\sum_{i=1}^{n} \quad \sigma_i{}^{-2})^{-1}$
- Integral of the product of normal PDFs
    - $\int_{-\infty}^{\infty} \quad N(x \mid \mu_1, \sigma_1{}^2)\, N(x \mid \mu_2, \sigma_2{}^2)\, dx \;=\; N(\mu_1 \mid \mu_2, \sigma_1{}^2 + \sigma_2{}^2)$

Readings

Barthelmé, S. (2016, March 22). *The Expectation-Propagation algorithm: a tutorial*

The first 16 minutes contains an overview of how the Expectation Propagation algorithm works. You apply this algorithm in the pre-class work.

The segment from 16:00 to 31:17 is optional.

Watch the "EP in one slide" segment from 32:17 to 35:15. Compare this slide to the description of EP in the study guide. If you didn't watch the previous optional segment, you won't understand what "natural parameters" are, but just think of them as the mean and variance for now. (They are actually the precision, 1/sigma^2, and the precision-adjusted mean, mu/sigma^2.)

Keep watching until 40:30 for some detail on why EP is a pretty fast algorithm.

Everything from 40:30 onwards is optional.

- KL-divergence
    - **Posterior distribution** of $\pi(\theta)$, which we wish to approximate with a Gaussian q
        - Goal: $arg\;\underset{q \in Q}{min} KL\;(\pi \;||\; q)$
        - $KL(\pi \;||\; q) = \int \quad \pi(\theta)\;\; log\,\frac{\pi(\theta)}{q(\theta)}\; d\theta$
    - **Posterior distribution** with n data points:
        - $\pi(\theta) \;=\; p(\theta \mid y) \;\propto\; p(\theta)\, \prod_{i=1}^{n} \quad likelihood(\theta) \;\propto\; \prod_{i=0}^{n} \quad likelihood_i(\theta)$
        - ⇒ `approximation using Gaussian factors:` $q(\theta) \;\propto\; \prod_{i=0}^{n} \quad q_i(\theta)$ (total of n+1 factors)
    - **Gaussian distribution**
        - Exponential representation: $q_i(\theta) = exp\;(-\frac{1}{2}\theta^T A_i \theta \;+\; r_i{}^T\, \theta)$ (quadratic to θ)
            - A_i: **precision** matrix
            - r_i: linear shift
        - Product of Gaussians:
            - $q(\theta) = \prod_{i=0}^{n} \quad q_i(\theta) = exp\left(-\frac{1}{2}\theta^T \sum_{i=0}^{n} \quad \{A_i\}\theta \;+\; \sum_{i=0}^{n} \quad \{r_i{}^T\}\theta\right)$
    - Hybridize true & approximate distribution
        - Take out approximate factor
            - $q_{-j}(\theta) = \prod_{i \neq j}^{n} \quad q_i(\theta)$ ⇒ **cavity distribution**
        - Insert **true factor** to create **hybrid**
            - $h_j(\theta) = l_i(\theta)\; q_{-j}(\theta)$ ⇒ **hybrid distribution**

Back to Top

        - **Project** the hybrid back to a Gaussian
            - **Normalizing factor** (b/c product of likelihoods): $z = \int \quad h_i(\theta)\, d\theta$
            - **Mean: Expectation** over **hybrids**: $E(\theta) = z^{-1} \int \quad \theta \cdot h_i(\theta)\, d\theta$
            - **Variance: covariance**: $\Sigma = z^{-1} \int \quad (\theta - E(\theta))(\theta - E(\theta))^T h_i(\theta)\, d\theta$
        - Update the approximate factor
            - Update q_j: find q_j such that $q_j q_{-j}$ has same moments as the **hybrid**
- Gaussians in exponential family of distributions
    - $q(\theta) \propto exp\,(-\frac{1}{2}\theta^T A_i \theta \,+\, {r_i}^T\,\theta)$
        - $=exp\,(\sum \quad A_{ij}\,(-\theta_i\theta_j/2) \,+ \sum \quad \theta_i r_i)$
        - $=exp\,(\,s(\theta)^T\lambda)$
            - $s(\theta)^T$: function of thetas (quadratic) ⇒ **sufficient statistics**
            - λ:for $\Sigma A_{ij}$and Σr_i⇒ **natural parameters**
    - Expressing Gaussians
        - **Natural** parameterization**: precision, shift**
            - $\lambda = \{A, r\}$
            - $q(\theta) \propto exp\,(-\frac{1}{2}\theta^T A_i \theta \,+\, {r_i}^T\,\theta)$
        - **Moment** parameterization: **mean, covariance**
            - $\eta = E_q(s(\theta))$
            - $q(\theta) \propto exp\,(\,s(\theta)^T\lambda)$
        - Transformation between natural to moment parameters
            - $\lambda = v(\eta)$
            - Precision = (covariance)$^{-1}$
            - Shift = precision * mean
    - Benefit of parameterization
        - **Multiplication** of sites: **addition** of natural parameters
            - $\prod \quad \{q_i(\theta)\} = \prod \quad exp\,(s(\theta)^T\lambda_i) = exp(s(\theta)^T \sum \quad \lambda_i$
        - **Taking out** a site (i.e. making cavities): **subtraction** of parameter
            - $q_{-j}(\theta) = q(\theta)/q_j(\theta) =exp\,(s(\theta)^T(\lambda - \lambda_j))$
- EP in one slide
    - Initialize site parameters $\lambda_1, \ldots, \lambda_n$
        - Global parameter: $\lambda = \Sigma\,\lambda_i$
    - When not converged, loop over i,
        - Form **cavity** $\lambda_{-j} = \lambda - \lambda_j$
        - Form **hybrid** $h_i(\theta) \,\propto\, l_i(\theta) exp(s(\theta)^T\lambda_j)$
        - Compute **moments** $\eta_j = E_{h_j}(s(\theta))$
        - **Transform** back to **natural** parameters $\lambda_j = v(\eta_j) - \lambda_{-j}$
        - Update **global approximation** $\lambda = \lambda_{-j}\lambda_j$
    - Computational cost (expensive operations):
        - Computing **moments** of the hybrid: $O(n)$for complete pass over data
        - Transforming natural parameters to moment parameters
            - ⇐ involves **matrix inverse**: $O(np^3)$ per complete pass over data
                - p: number of parameters,
                - For small p, n dominates

[Back to Top](#)

- ⇐ in typical problems, EP **stabilizes** after 3-4,
    - ⇒ so $O(n)$is non-scaling!!

Sutton (2005). *Expectation propagation in factor graphs: A tutorial*.

This tutorial provides another overview of the Expectation Propagation, pointing out that there are two interpretations of how the algorithm functions — as message passing on a factor graph and as a projection of the true probability distributions in the model onto exponential family distributions (in our case, normal distributions). Read the first 3 sections. The fourth section is optional.

-

Lambert (2018, May 15). *Explaining the Kullback-Leibler divergence through secret codes*

[10 minutes] A very simple explanation of how the Kullback-Leibler divergence measures by how much two probability distributions differ.

- KL-divergence
    - $KL(P(x) \rightarrow Q(x)) = \sum_x \quad \{P(x)\, log(P(x)/Q(x))\}$
    - **Lower bound** on informational cost

(Optional) Section 10.7, Bishop (2006) Pattern recognition and machine learning

Bishop gives a detailed guide to the Expectation Propagation algorithm as well as how it is used in factor graphs. Focus on how to update approximate factors by calculating sufficient statistics (mean and variance). The section on using Expectation Propagation in factor graphs (10.7.2) is particularly important since you will see how that is applied to a real-world problem in Session 11.2.

Class

Back to Top

CS146 10.2 - Synthesis: Verifying the GPS tracking model

Pre-class

https://gist.github.com/AntonioStark/59ba350652ecf791a84f58ed3f818a59

Study guide

- **Model checking**
  - Use **posterior predictives**
  - ⇒ p-values, histograms, scatterplots
  - ⇐ any model can be **fit** to data, but doesn't say the model is **good**
- Model checking methods
  - **Replicated data:** should have same **distributional properties** to real data
  - **External validation**: test predictions using new data (but new data is expensive)
- Model **checking != comparison**
  - **Model comparison**: quantifies how well **each model** explains the data
  - **Model checking**: find the **deficiencies** of a particular model
- **Posterior predictive checks**
  - **Posterior**: P(parameters | real data)
  - ⇒ generate replicate data: **likelihood**: P(replicated data | parameters)
  - **Posterior predictive p-values**:
    - **0.5**: distribution of test statistics from replicated data is exactly centered on test statistic of real data
    - **0**: entire replicated test statistic is to the left (less than) real test statistic value
    - **1**: entire replicated test statistic is to the right (greater than) real test statistic value
    - Gelmen reading: doubt if test statistic is outside of 90% confidence interval
- **Test statistic** from **data** only:
  - Sample $\theta_i \sim p(\theta_i \mid d)$, and generate replicated dataset with **same size** to real data set (⇒ test statistic function has same statistical properties over the replicated & real datasets) $d_i^{rep} \sim p(d \mid \theta_i)$
  - Compute test statistic on replicated dataset $T(d_i^{rep})$
  - With enough samples, **histogram** of **replicated test statistic values** $\{T(d_i^{rep})\}$will approximate the **distribution** of replicated test statistic
- **Test quantity** from data (observed variables) + **parameters** (unobserved variables):
  - Any function of $T(d, \theta)$
  - Generate sample from posterior distribution $\theta_i \sim p(\theta_i \mid d)$
  - Generate **replicated dataset** (again with same size) $d_i^{rep} \sim p(d \mid \theta_i)$
  - Compute test quantity using **real** dataset (x coordinate): $T(d, \theta_i)$
  - Compute test quantity using **replicated** dataset (y coordinate): $T(d_i^{rep}, \theta_i)$
  - **P-value**: $T(d_i^{rep}, \theta_i) > T(d, \theta_i)$

Back to Top

Readings

Chapter 6, Gelman (2013) Bayesian data analysis

Read Sections 6.1–6.5. The rest of the chapter is optional. You might find it helpful to read the study guide before reading this Gelman chapter. Focus on the different ways in which we can evaluate whether a model explains a data set well and on the interpretation of p-values, histograms, and scatter plots such as those in Figure 6.5 on page 164.

-

(Optional) Lambert (2014) Prior and posterior predictive distributions — An introduction

This video is optional and reviews what posterior predictive distributions are. This could be a useful refresher since posterior predictive distributions are used to calculate posterior predictive test statistics.

-

Lambert (2014) What is a posterior predictive check and why is it useful?

This video explains and provides an example of computing posterior predictive checks, based on the Gelman reading. Focus on understanding how samples generated from the posterior distribution are used to compute a "Bayesian p-value"

- **Posterior predictive distribution**:
    - Probability of new data over posterior distribution
    - $p(\tilde{x} \mid x) \approx \theta_i \sim p(\theta \mid x) + \widehat{x_i} \sim p(\hat{x} \mid \theta_i)$

Class

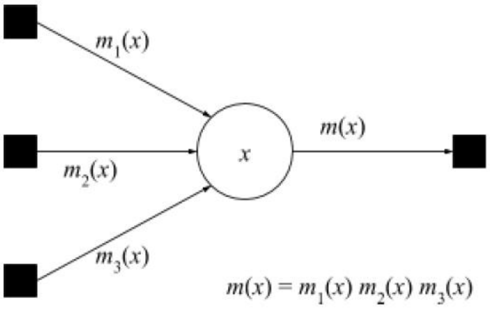
Back to Top

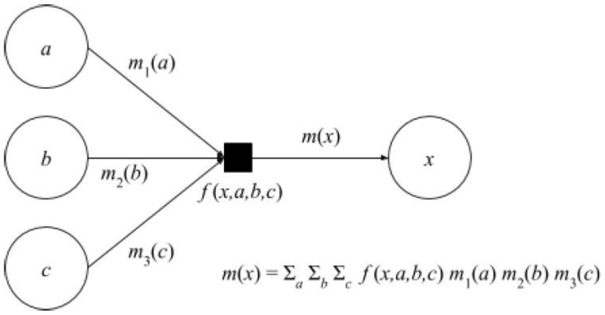
CS146 10.1 - Message passing on cyclic graphs

Pre-class

https://docs.google.com/document/d/1Qimv3ktGAUOXSVpMvhjL7MZx4JvX7AjsjKtmq7Zlf1s/edit?usp=sharing

Study guide

- **Sum-product algorithm**
  - Efficiently computes marginal distributions from joint distribution
  - ⇐ utilizes the structure of **factor graphs** that represent the joint distribution
- **Efficiency** of sum-product
  - Before: O(k^{n-1}) (when there are n terms, each with k possible discrete values)
  - Now: O(max(k^{i-1},k^{n-i})) (where i denotes the element of marginalization)
  - $$\begin{aligned} p(x_i) &= \sum_{x_1} \sum_{x_2} \cdots \sum_{x_{i-1}} \sum_{x_{i+1}} \cdots \sum_{x_{n-1}} \sum_{x_n} p(x_1, x_2, \cdots, x_n) \\ &= \sum_{x_1} \sum_{x_2} \cdots \sum_{x_{i-1}} \sum_{x_{i+1}} \cdots \sum_{x_{n-1}} \sum_{x_n} p(x_1, x_2, \cdots, x_i) p(x_i, x_{i+1}, \cdots, x_n) \\ &= \sum_{x_1} \cdots \sum_{x_{i-1}} p(x_1, x_2, \cdots, x_i) \cdot \sum_{x_{i+1}} \cdots \sum_{x_n} p(x_i, x_{i+1}, \cdots, x_n) \end{aligned}$$
  - $$\begin{aligned} p(c) &= \sum_{a=1}^{10} \sum_{b=1}^{10} \sum_{d=1}^{10} \sum_{e=1}^{10} p(a,b,c,d,e) \\ &= \sum_{a=1}^{10} \sum_{b=1}^{10} \sum_{d=1}^{10} \sum_{e=1}^{10} p(a|b)p(b|c)p(c|d)p(d|e)p(e) \\ &= \sum_{a=1}^{10} \sum_{b=1}^{10} p(a|b)p(b|c) \cdot \sum_{d=1}^{10} \sum_{e=1}^{10} p(c|d)p(d|e)p(e) \end{aligned}$$
- **Message** passing
  - Variable → factor: product of all other messages
    - 

  - Factor → variable: product(factor, incoming messages) {marginalized by all incoming variables)
    - 
a
$m_1(a)$
$m(x)$
b
x
$m_2(b)$
$f(x,a,b,c)$
$m_3(c)$
c
$m(x) = \Sigma_a \Sigma_b \Sigma_c\, f(x,a,b,c)\, m_1(a)\, m_2(b)\, m_3(c)$
- **Marginal distribution**:

[Back to Top](#)

  - Compute by taking the product of the **messages going into** the variable of interest
- Integration of product of normals
  - $\int_x \quad N(x \mid \mu_1, {\sigma_1}^2) N(x \mid \mu_2, {\sigma_2}^2) = N(\mu_1 | \, \mu_2, {\sigma_1}^2 + {\sigma_2}^2) = N(\mu_2 | \, \mu_1, {\sigma_1}^2 + {\sigma_2}^2)$

Readings

Sections 8.4.6–8.4.8, Bishop (2006) Pattern recognition and machine learning

This is a short reading on approaches to doing inference in graphs with loops. Read all of it as general background, but focus on Section 8.4.7 on loopy belief propagation. This is the method that we will explore in the pre-class work and during class.

-

Zabaras (2018) The sum-product algorithm on factor-trees

This video provides a review and examples of applying the sum-product algorithm in various factor graphs. Segments marked with an asterisk (*) are required.
(*) Watch up to 0:11:00 for a review of directed graphs and loopy message passing on directed graphs.
(*) 0:11:00–0:20:30 A review of factor graphs.
0:20:30–0:45:00 A very detailed description of the sum-product algorithm on factor graphs.
(*) 0:45:00–0:47:30 Initializing the sum-product algorithm on factor graphs.
0:47:30–0:58:00 Example 1, which is similar to our GPS model.
0:58:00–1:08:40 Example 2 of using factor graphs.
1:08:40–1:10:00 Example 3 of using factor graphs.
(*) 1:10:00–1:15:30 How to compute marginals using messages in the sum-product algorithm.

-

(Optional) Chapter 6, Barber (2012) Bayesian Reasoning and Machine Learning

The junction tree algorithm is used for running the sum-product algorithm in finite time (without iteration) on graphs with loops. We will not cover the junction tree algorithm in this course, but if you are interested in understanding efficient methods for exact inference on graphical models further, this chapter is a good place to start.

-

(Optional) Bromiley (2014) Products and Convolutions of Gaussian Probability Density Functions

See the study guide for a summary of how to work with products of normal pdfs. We see a lot of normal products in class and the pre-class work, since messages on factor graphs are formed from products of other messages. This reading is optional and provides a detailed proof of the product of normals identity given in the study guide.

-

Back to Top

Class

Back to Top

CS146 9.2 - Sum-product algorithm

Pre-class

gDoc: https://docs.google.com/document/d/1NvalK1-QRYmTA9A-8VHEatDcJUgAMBjW-X86Ir9oP18/edit?usp=sharing

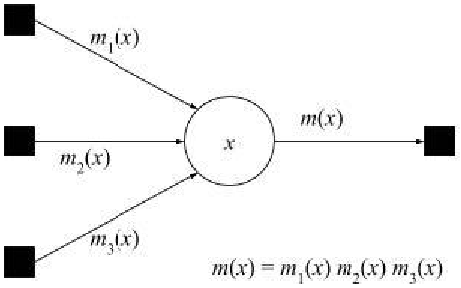
Study guide

- **Sum-product algorithm**
  - Efficiently computes **marginal distributions** from **joint distributions**
  - "Normal" marginalization: requires $k^{n-1} evaluations$ (⇒ **exponential** computation) for a graph of *n* variables, each with *k* states
- **Marginal distribution**
  - $p(x_i) = \sum_{x_1} \cdots \sum_{x_{i-1}} \sum_{x_{i+1}} \cdots \sum_{x_n} p(x_1, x_2, \cdots, x_n)$

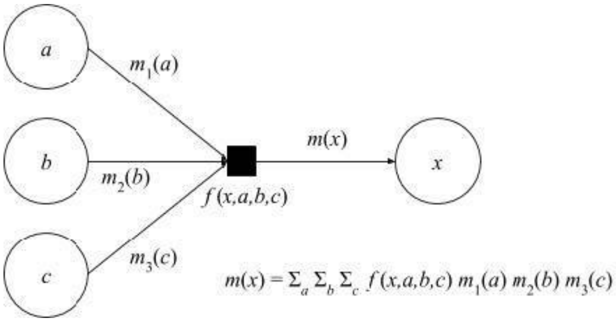
    $$= \sum_{x_1} \cdots \sum_{x_{i-1}} \sum_{x_{i+1}} \cdots \sum_{x_n} p(x_1, x_2, \cdots, x_{i-1}, x_i) p(x_i, x_{i+1}, \cdots, x_n)$$

    $$= \sum_{x_1} \cdots \sum_{x_{i-1}} p(x_1, x_2, \cdots, x_i) \sum_{x_{i+1}} \cdots \sum_{x_n} p(x_i, x_{i+1}, \cdots, x_n)$$
  - Now # of calculation is $max(k^{i-1}, k^{n-i})$ instead of k^{n-1}
  - $p(c) = \sum_{a=1}^{10} \sum_{b=1}^{10} \sum_{d=1}^{10} \sum_{e=1}^{10} p(a, b, c, d, e)$

    $$= \sum_{a=1}^{10} \sum_{b=1}^{10} \sum_{d=1}^{10} \sum_{e=1}^{10} p(a|b)\, p(b|c)\, p(c|d)\, p(d|e)\, p(e)$$

    $$= \sum_{a=1}^{10} \sum_{b=1}^{10} p(a|b)\, p(b|c) \sum_{d=1}^{10} \sum_{e=1}^{10} p(c|d)\, p(d|e)\, p(e)$$
- Messages
  - Always between factor or variable nodes, one each
  - Each message: function of **exact one variable** (the variable being the variable which the message is sent to/from)
  - 

Back to Top

- 
a
b
c
x
$m_1(a)$
$m_2(b)$
$m_3(c)$
$m(x)$
$f(x,a,b,c)$
$m(x) = \Sigma_a \Sigma_b \Sigma_c f(x,a,b,c)\, m_1(a)\, m_2(b)\, m_3(c)$

Readings

Section 8.4.4, Bishop (2006) Pattern recognition and machine learning

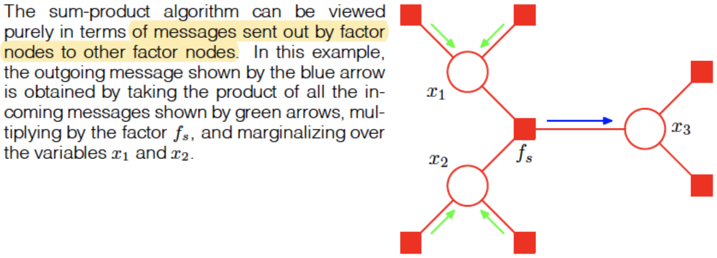
This section describes the sum–product algorithm in detail. The mathematical derivation can be a lot to absorb on a first reading, so use the videos below to help you understand how the algorithm works. The purpose of the sum–product algorithm is to compute the marginal distributions over all variables in a factor graph.

Optionally, you can also read Section 8.4.5, which describes the related max–sum algorithm. The sum–product algorithm computes the marginal distribution over each variable in the graph. The max–sum algorithm computes the most probable value (the mode) of the joint distribution over all variables. Note that the sum–product algorithm does not give you the mode, since the marginal mode of each variable might be quite different from the joint mode over all variables. We will not discuss the max–sum algorithm in class.

- **Sum-product algorithm**
  - Efficient, exact inference algorithm
  - Applicable to **tree-structure** graph
  - Marginalization: sums (discrete variables) or integrals (linear)
- **Messages:** always a function of the **variable** associated with the **variable node** that the link connects to
  - Evaluate `factor → variable` message: {(product of incoming messages along all other links coming into the factor node)*(factor associated with the node)}[marginalize overall variables associated with the incoming messages]
  - Evaluate `variable → factor` message: product(incoming messages along all other links)
  - **Variable nodes** with only **two neighbors**: perform no computation, but pass message unchanged
- **Marginal** (of a variable):
  - Product of incoming messages along all links arriving to that node
  - **Root node**: the variable whose marginals are being calculated
  - **Leaf node**: end nodes
    - **Variable leaf node**: $\mu_{x \to f}(x) = 1$
    - **Factor leaf node**: $\mu_{f \to x}(x) = f(x)$
- **Validity**: each node always **receives** enough messages to **send out** a message. **Proof by induction** of adding new leaf nodes
- **Efficiency**:

Back to Top

  - # of computation required to calculate all messages in graph = twice the # of links in the graph = twice the computation of finding a single marginal
- Perspective: graph consists only of factor → variable messages



The sum-product algorithm can be viewed purely in terms of messages sent out by factor nodes to other factor nodes. In this example, the outgoing message shown by the blue arrow is obtained by taking the product of all the incoming messages shown by green arrows, multiplying by the factor f_s, and marginalizing over the variables x_1 and x_2.

  - b/c variable → factor messages are the product of incoming messages

Bishop (2013). Graphical Models 2

This is part 2 of a 3-part video series on graphical models, which you started watching for the previous session. Watch 1:03:50–1:24:00 on efficient inference and the sum–product algorithm. The focus of today's session is understanding the sum–product algorithm.

- **Efficient inference**
  - By exploiting **factorization**, I can *switch out* the sums of products into a product of sums, which is more computationally efficient
  - $\sum_x \sum_y xy = x_1y_1 + x_1y_2 + x_2y_1 + x_2y_2 = (x_1 + x_2)(y_1 + y_2)$
- **Sum-product algorithm**
  - (1) The **marginal** of a particular **variable**: **multiply** all the **incoming messages** from neighboring factor nodes
  - (1) $p(x) = \prod_{f \in F_x} m_{f \to x}(x)$
  - (2) **message** from **factor node → variable node**: {(product of all incoming messages from neighboring **variable nodes**) * (factor at each factor node)} sum over all other (i.e. not outgoing) variables
  - (2) $m_{f \to x_1}(x_1) = \sum_{x_2} \sum_{x_3} \cdots \sum_{x_n} f(x_1, x_2, x_3, \cdots) \prod_{i>1} m_{x_i \to f}(x_i)$
  - (3) **message** from **variable node → factor node**: product of all other incoming messages
  - (3) $m_{x \to f}(x) = \prod_{f_j \in F_x \setminus \{f\}} m_{f_j \to x}(x)$
  - Only works for **tree-structure** graphs (i.e. a single node breaks the tree into two parts)
    - If there are **loops**, sum-product algorithm won't work
  - **All the marginals** need only twice as much work as computing the marginal of a single variable, as all factors have been **computed** (which can then be **cached**)

(Optional) Bishop (2013) Graphical Models 3

This is part 3 of a 3-part video series on graphical models. This video discusses the Kalman filter for solving a problem **very similar to the GPS tracking problem** we have been working on. The important part today, is to understand the sum–product algorithm, as discussed in the previous video (Part 2). This part is optional for those who would like to learn more about the overlap between the Kalman filter and the sum–product algorithm on factor graphs. Watch 0:00:00–0:38:40.

- **Kalman filter**: Gaussian (radar)

Back to Top

  - Backwards in time: Kalman smoother
- **Hidden Markov model**: discrete (speech recognition)
- Stationary hand: three sources of uncertainty
  - 1. Noise of the sensor
  - 2. Position of the hand (variance of μ)
  - 3. Value of the noise (variance of λ)
  - The latter two uncertainties go away with more data

(Optional) Hamprecht (2012, November 22) 10.3 Message Passing | 10 Directed Graphical Models | Pattern Recognition Class 2012

The first part of this video lecture is worth watching. It provides a nice review of factor graphs and provides a simple example of message passing in factor graphs in the first 17 minutes of the video. There is also a full derivation of the sum-product algorithm up to 33:30.

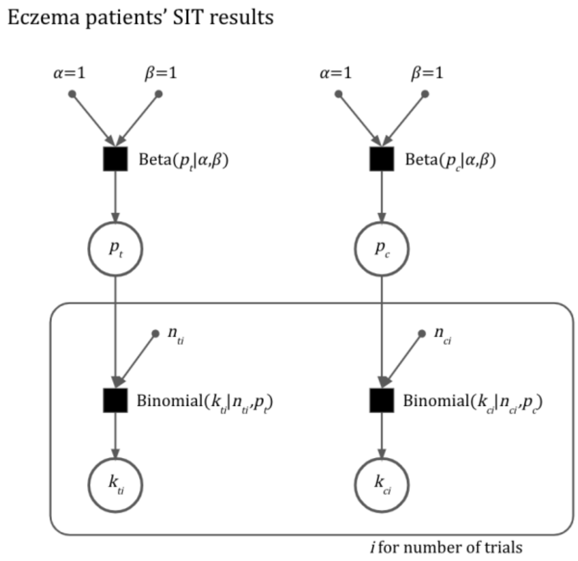
Class

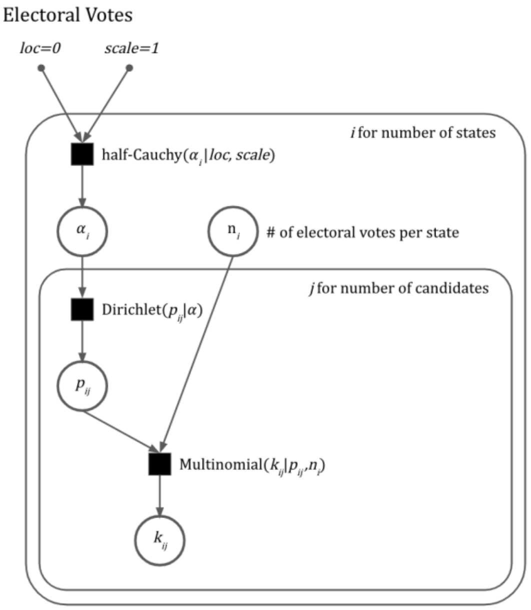
Back to Top

CS146 9.1 - Factor Graphs

Pre-class

GitHub: https://gist.github.com/AntonioStark/5912811c9c95fd7ee637705cf5fb123d




Electoral Votes
$loc=0$
$scale=1$
half-Cauchy($\alpha_i|loc, scale$)
i for number of states
α_i
n_i
# of electoral votes per state
j for number of candidates
Dirichlet($p_{ij}|\alpha$)
p_{ij}
Multinomial($k_{ij}|p_{ij},n_i$)
k_{ij}

Study guide

- Purposes of **factor graphs**
  - 1. Represent **probabilistic models**: form of **notation**
  - 2. Conduct **inference**: a computational tool
- Factor graphs
  - **Variable** nodes (circles)
  - **Factor** nodes (filled black square)
  - **Plates** (repetition)
  - **Full joint probability distribution** of model, over all parameters & data: the product of all the **factors** (square nodes) in the graph
- Directed graphs vs. factor graphs

Readings

Section 8.4.3, Bishop (2006) Pattern recognition and machine learning

This section introduces factor graphs and their properties. Today's session is about reading, drawing and understanding how to sample from factor graphs, so focus on these aspects.

- Global function of several variables: product of factors over subsets of the variables

Back to Top

Bishop (2013) Graphical Models 1

This is part 1 of a 3-part video series on graphical models. The relevant video starts at 1:00:00 (1 hour) and ends at 1:15:00, covering directed graphs. This part of the video summarizes our discussion on directed graphs from the previous class. The first part of the video (up to 1:00:00) is for general interest. It is worth watching if you would like to understand the connection between machine learning and statistical modeling better.

- Definitions of probability
  - Limit of an infinite number of trials
  - Quantification of **uncertainty**
- **Factor graph**
  - Circle: **variable node**
  - Square box: **probability distribution**
- P(x,y) = P(y|x)P(x)
  - P(x,y): **joint** distribution
  - P(y|x): **conditional** distribution
  - P(x): **marginal** distribution
- **Marginalization**: sum rule
- Rules of probability
  - **Sum rule**
    - $P(x) = \sum_y \quad P(x, y)$
  - **Product rule**
    - $P(x, y) = P(y|x)\, P(x)$
  - **Bayes' theorem**
    - $P(y|x) = \frac{P(x|y)\, P(y)}{P(x)}$
  - **Denominator (marginalization)**
    - $P(x) = \sum_y \quad P(x|y)\, P(y)$
- Three types of graphical model
  - 1. **Directed graphs**
    - Useful for designing models
    - Represent a **family** of distributions according to a specific **factorization rule**
  - 2. **Undirected graphs**
    - Good for some domains like computer vision
  - 3. **Factor graphs**
    - Useful for inference & learning

Bishop (2013) Graphical Models 2

This is part 2 of a 3-part video series on graphical models. The relevant video starts at 0:00:00 and ends at 1:04:00. Watch 0:00:00–0:30:50 on conditional independence in directed graphs. You can skip 0:30:50–0:59:00 on undirected graphs since we will not be discussing this at all (but it is optional viewing for those who are keen). Watch 0:59:00–1:03:50 on factor graphs. Today's session is about reading, drawing and understanding how to sample from factor graphs, so focus on these topics in the video.

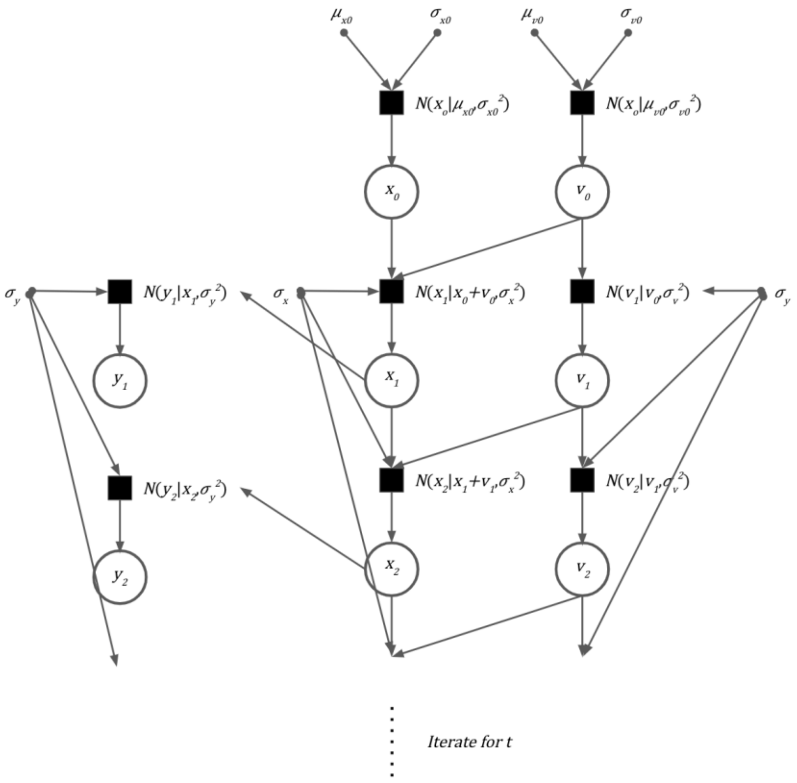
- **Conditional independence**
  - When $P(a, b) = P(a)P(b)$
  - I.e. when nodes a and b are **unconnected**

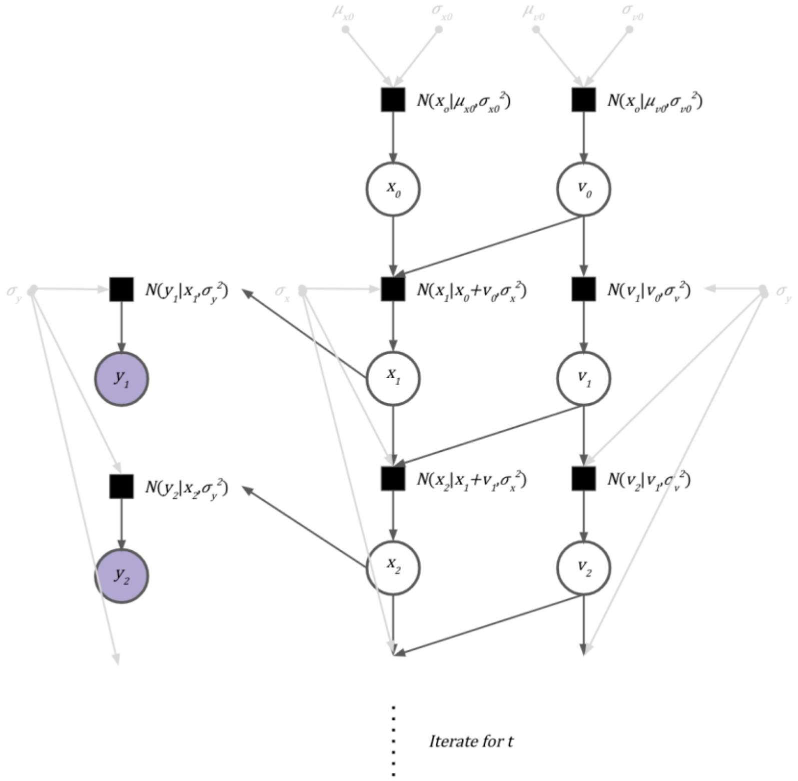
Back to Top

  - Proof: conditional probability $P(a|b) = \frac{P(a,b)}{P(b)} = P(a)$
  - The information of b is uninformative to variable a
  - For conditional on another variable,
  - $P(a, b \mid c) = P(a|b, c)P(b|c) = P(a|c)\, P(b|c)$
  - **Tail-to-tail conditional independence** (the middle node's observation "blocks" the path)
  - **Head-to-tail conditional independence** (the middle node's observation "blocks" the path)
  - **Head-to-head independence** & **conditional dependence** (the middle node's observation "unblocks" the path)
- **D-seperation criteria**
  - Find out "paths" between the two nodes
  - And find out if any part is "blocked"
- Factor graphs
  - Joint distribution = product of the factors

Back to Top

Class





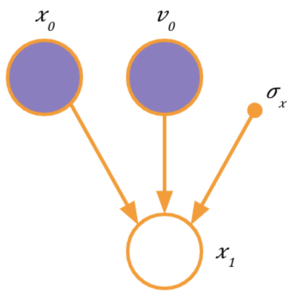
Back to Top

CS146 8.2 - Directed Graphical Models

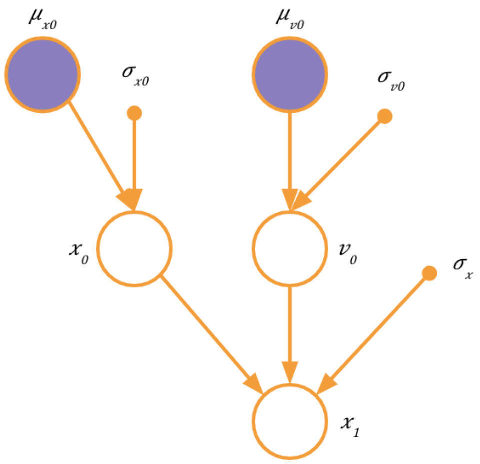
Pre-class

Google slides: CS146 8.2 Preclass Antonio

Task 1



Task 2



Study guide

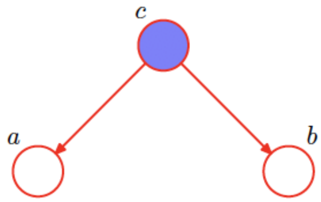
- Probability theory:
  - **Sum** rule + **product** rule
- **PGM: probabilistic graphical model**: represent statistical models
  - **Node:** variable (or collection thereof)
    - **Empty circle**: unknown/unobserved variable
    - **Filled circle**: known/observed (data)
    - **Small dot** (and variable name): **hyperparameter** (known quantity but **not data**)
    - **Rectangle** (plate): repetition (for-loops)
  - **Edge**: dependence between nodes. Arrow indicates dependence
- **Generative probability model**
  - Generate samples via **ancestral sampling** mechanism

Back to Top

Readings

Sections 8.1–8.3, Bishop (2006) Pattern recognition and machine learning

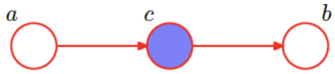
This reading covers the same material as the series of videos below. You may choose to focus on the videos, to focus on this reading, or to use both resources to complement each other.
You should focus on the following sections for class: 8.1.1, 8.1.2, 8.2.1. The remaining sections are optional. We will not cover undirected graphs (Section 8.3) in any detail but move straight on to factor graphs in the next session.
For today's session conditional independence, factorization, and how to represent a generative model as a directed graphical model are all important concepts/topics. Directed graphical models provide a new type of notation and it is very important to understand the value of this notation. Generally, good notation guides and supports our understanding of a model.

- **Joint distribution** defined by a graph:
    - Given as a product over all the nodes in the graph, of conditional distribution for each node, conditional on the variables corresponding to the parents of that node
    - $p(X) = \prod_{i=1}^{N} \quad p(x_k \mid pa_k)$
        - $X = \{x_i \, for \; all \; i\} = \{x_1, \ldots, x_N\}$
        - Pa$_k$: set of parents of x$_k$
        - ⇒ **factorization properties** of joint distribution for a directed graphical model
- **DAG: Directed acyclic graphs**
    - Has no **directed cycles** (no directed closed paths within the graph)
- Notation
    - **Latent** variable (**hidden** variable): value is not observed
- **Generative models**
    - **Ancestral sampling**
    - Higher-number variables: terminal nodes: **observations**
    - Lower-number variables: **latent variables**
        - ⇒ allow models to be constructed from simpler conditional distributions
    - Represent the **causal process** by which observed data is generated
- Conditional independence
    - 

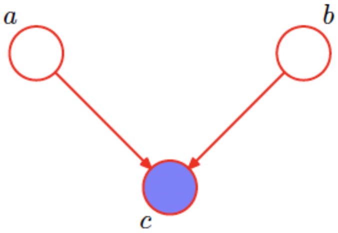
        - $p(a, b, c) = p(a|c)\, p(b|c)\, p(c)$
        - C is **tail-to-tail** with respect to path from a to b
    - Independence between a and b (⇒ NOT independent)
        - $p(a, b) = \sum_c \quad p(a|c)\, p(b|c)\, p(c) \;\neq p(a)\, p(b)$
    - **Conditional** independence between a and b, conditional to c (⇒ conditionally INDEPENDENT)
        - $p(a, b|c) = \frac{p(a,b,c)}{p(c)} = p(a|c)\, p(b|c)$

Back to Top

        - $\therefore a \perp\!\!\!\perp b \mid c$
        - Conditioning of c **blocks** the path, and a,b become independent
    - Also works for:
        - **Head-to-tail**



    - Works in **reverse** for **head-to-head**
        - a and b are **generally independent**, but **conditionally dependent**



        - **Head-to-head path** becomes **unblocked** if either the **node** or any of its **descendants** are observed

[mathematicalmonk] (2011) Machine Learning

This series of videos cover the same material as the Bishop reading above. You may choose to focus on the videos, to focus on the Bishop reading, or to use both resources to complement each other.
Note that some of the videos in the series are optional, namely 13.6, 13.7, 13.10, 13.11, 13.12, 13.13. All videos listed below are required viewing (unless you read the Bishop chapter instead)
For today's session conditional independence, factorization, and how to represent a generative model as a directed graphical model are all important concepts/topics. Directed graphical models provide a new type of notation and it is very important to understand the value of this notation. Generally, good notation guides and supports our understanding of a model.

- **Directed Graphical Models** (a.k.a. **Bayesian networks** or **conditional independent diagrams**)
    - Graphical: notational device
        - ⇒ visualize **conditional independence** properties
            - ⇒ if conditionally independence: **tractable inference**
        - ⇒ visualize **inference algorithms** for a particular probability distribution (dynamic programming, markov-chain monte carlo)
    - **DAG: directed acyclic graph**:
        - Edges are oriented (directed)
        - There are no directed cycles (there is always an order for an edge)
    - Formalism
        - pa(i): parents of node i
        - x_A: set of node x_i of i ∈ A
        - Given *X*=(X_1, … , X_n) ~ *p* (PMF or PDF) and an ordered DAG *G* on *n* vertices, we say **X respects G** (or **p respects G**) if $p(x_1, \ldots, x_n) = \prod_{i=1}^{n} \quad p(x_i \mid x_{pa(i)})$
        - ⇒ does **not** imply conditional **dependence** between variable (only implies conditional **independencies**)
        - Any **distribution** respects any **complete DAG**
    - **Complete directed graph**: where there is an edge between any and all two vertices

Back to Top

    - Can combine random variables into vectors
    - If the factors are **normalized conditional distributions**, then the **product** is a normalized probability distribution as well
- **Generative Process Specification**
    -
- Conditional Independence in Graphical Models

Class

Back to Top

CS146 8.1 - Generative Models

Pre-class

In a generative model, we model the field goal rate by any professional player p(y) using a Beta distribution, but with unknown parameters \alpha and \beta. Our initial guess for the parameters are 1's (i.e. the uninformative prior) but our estimate for them evolves as we observe the three players (three samples from the population of professional players). Using the binomial likelihood function and our data of attempts \n and successes \k for each player, we generate posterior parameters for our beta distribution. These posterior parameters (and the Beta distribution) now define the generative model for the field goal rate for the new (and any) professional player. This is different from the discriminative model whose prior model for the new player will be the uninformative prior (i.e. 1's)

Model

- X: probability of success (field goal rate)
- Y: number of successful throws from n attempts
- Posterior p(y|x): Beta distribution
- Prior p(

Study guide

- This session synthesizes what you have learned so far about data modeling by rethinking data modeling from the generative perspective.
- Any fully defined probability model also describes how data are generated by the model — or possibly even how data are generated in the real world, depending on how closely the model approximates reality.
- Focus on thinking about how a given data set could have been generated in the modeling sense — what are the relevant variables (parameters, hyperparameters, data) and from which distributions were they generated or sampled?

Readings

Ng (2015). Naive Bayes generative learning algorithms

Focus on the interpretation of a probability model as a generative model. Generative modeling is a conceptual framework for how we work with data sets — viewing observed data as samples from some well-defined probability model. "Generating from a model" is the process of producing samples (data) from the model. "Bayesian inference" is the process of applying Bayes rule to calculate posteriors over parameters of the model once we observed data from the real world (not samples for the model).

- Categories of ML algorithms
    - 1. **Discriminative** learning algorithms
        - Learn p(y|x) directly (i.e. y as a direct function of x)
            - x: feature
            - y: category/label

Back to Top

        - Ex) Logistic regression, any other that use **gradient descent**
    - 2. **Generative** learning algorithms
        - Learn p(x|y) and p(y)
            - p(x|y): given our label, what would the data look like
        - Using Bayes' Rule (on two-label classification)
            - $p(y=1|x) = \frac{p(x|y=1)\,p(y=1)}{p(x)}$
            - Where $p(x) = \sum_y \quad p(x,y) = p(x|y=1)p(y=1) + p(x|y=0)p(y=0)$
        - Build **models** of classes (from training sets)
        - Then compare new data points to the models

Scheffler (2018) Generative probability models

This reading covers the main concepts on generative modeling needed to follow this session. Make sure you are comfortable with all concepts in these notes.

- Eczema case study
    - Notation:
        - # of patients in each group: n_t, n_c
        - # of improved patients in each group: k_t, k_c
        - Probability of improvement in each group: p_t, p_c
        - Prior hyperparameters: α, β
    - Bayes components
        - **Likelihood:** $p(data \mid parameters) \; = \; Binomial(k_t \mid n_t, p_t)\; Binomial(k_c \mid n_c, p_c)$
        - **Prior**: $p(parameters) \; = \; Beta(p_t \; \mid \; \alpha, \beta)\; Beta(p_c \mid \alpha, \beta)$
- Generative interpretation
    - For a yet unknown RCT → modeling the probabilities (beta) and number (binomial) of success
    - $P(k_t, k_c, p_t, p_c \mid \alpha, \beta, n_t, n_c) \; = \; Binomial(k_t \mid \; n_t, p_t)\; Binomial(k_c \mid n_c, p_c)\; Beta\;(p_t \mid \alpha, \beta)\;\; Beta\;(p_c \mid \alpha, \beta)$

Ng, A.Y. (n.d.) Generative Learning algorithms

For those who are interested in the application of probability theory to Machine Learning, these notes from Andrew Ng's course at Stanford describes the use of generative modeling in machine learning tasks.

-

(Optional) Ng (2001) On discriminative vs generative classifiers: A comparison of logistic regression and naive Bayes

This article describes what is briefly discussed in today's video by Andrew Ng — comparing discriminative and generative models for classifications tasks. While the discriminative approach achieves better accuracy when a lot of data are available, the generative approach achieves better accuracy when the data set is relatively small (it converges more quickly).

-

Back to Top

Class

- **Discriminative** model: find **posterior probability** p(y|x)
  - ⇒ models the **decision boundary** between classes
  - ⇐ assumes functional form for p(y|x)
  - ⇐ and estimate parameters of p(y|x) directly from training data
  - Examples
    - Logistic regression
    - Support vector machine (SVM)
    - Traditional neural networks
    - Nearest neighbor
- **Generative** model: find **joint probability** p(x,y)
  - ⇒ models the **actual distribution** of each class
  - ⇐ assumes functional forms for p(y) p(x,y)
  - ⇐ estimate parameters of p(x|y), p(y) from training data
  - ⇐ use **Bayes rule** to calculate p(y|x)
  - Examples
    - Naive Bayes
    - Bayesian networks
    - Hidden Markov Models (HMM)

Back to Top

CS146 7.2 - Modeling hierarchical survey data

Pre-class

GitHub: https://gist.github.com/AntonioStark/13618899594c3670d8cf3afee3491cc7

- ‘Survey_results’: dictionary: states to survey results
    - Row: one survey
    - Column: one candidate (Clinton, Trump, Johnson, Stein)

Study guide

Readings

ABC News. (2016, November 6). *Nate Silver on FiveThirtyEight's election day forecast* [Video file].

Retrieved from https://www.youtube.com/watch?v=gYi6Ibf3e0o
In the first session of this course, we watched a video on the Fivethirtyeight.com predictions of the 2016 US presidential elections. This video is an interview with Nate Silver containing a more nuanced discussion of how poll data are integrated into the Fivethirtyeight.com model for making predictions. Listen to the explanation of why the distribution of votes between states matters. Considering only the national poll numbers hides a lot of important details.

Grey, C.G.P. (2011, November 7). *How the electoral college works* [Video file].

Retrieved from https://www.youtube.com/watch?v=OUS9mM8Xbbw
This video will help you further understand why the electoral college votes, which are influenced by state polls rather than national polls, are important. You should watch up to at least 1:40 for the simple explanation of how voting works in the USA. The remaining 3 minutes are optional, and provide interesting information on how approximately 3% of the population are excluded from voting.

Silver, N. (2016, Jun. 29). *A user’s guide to FiveThirtyEight’s 2016 general election forecast.*

Retrieved from https://fivethirtyeight.com/features/a-users-guide-to-fivethirtyeights-2016-general-election-forecast/
(Optional) This is a long blog post on how the real FiveThirtyEight model was developed and used to forecast the outcome of the 2016 USA general elections. Read the post to get a sense of how much effort goes into developing a detailed probabilistic model for predictions in the real world.

-

Back to Top

Class

Back to Top

CS146 7.1 - Central Limit Theorem

Pre-class

GitHub: https://gist.github.com/AntonioStark/34d576867e8cb895d4277f930b51044f

Study guide

- **Central Limit Theorem (CLT)**
  - ΣX for X that is **i.i.d. (independent & identically distributed**) randomly distributed will follow a **normal distribution** as n → ∞
  - ⇒ identify if the quantity of approximation is the **sum** or **linear transformation** of the sum of **i.i.d. Random values**
  -

Readings

Balka, J. [jbstatistics]. (2012, December 28). *Introduction to the Central Limit Theorem* [Video file].

Retrieved from https://www.youtube.com/watch?v=PujoI1yC1_A
This video demonstrates the Central Limit Theorem (CLT) using sampling and visualization. Focus on building your intuition for why sample estimates approximately follow a normal distribution, even when the underlying distribution being sampled is not even close to normal. Also, pay careful attention to the example starting around 8:30 in the video — it demonstrates how easy it can be to apply the CLT incorrectly.

- **CLT**: the distribution of the **sample mean** tends to the normal distribution as **sample size** increases, regardless of the distribution of the population
  - ⇒ **sample mean** is roughly normally distributed for samples whose n >= 30
  - $\frac{\bar{X}-\mu}{\sigma/\sqrt{n}} \to (converge\ to)\ N(0,1)\ as\ n \to \infty$ as long as mean and variance are finite
  - Z-score: $\frac{x-\mu}{\sigma/\sqrt{n}}$ (z-score is unrelated to normal distribution)

Keng, B. (2016, April 2). Normal approximation to the posterior distribution. *Bounded Rationality*

Retrieved from http://bjlkeng.github.io/posts/normal-approximations-to-the-posterior-distribution/
This is an in-depth reading on the Central Limit Theorem in Bayesian data analysis. The proofs are optional. Focus on the motivation behind using a Gaussian to approximate posterior distributions. The point here is **not that any distribution can be approximated by a Gaussian**, but that **posterior distributions** can be if we have enough data. See the study guide for notes on the KL-divergence if you have never heard of it before.

- Bayes' theorem

Back to Top

    - $P(\theta|y) = p(y|\theta)\, P(\theta) \,/\, P(y)$
        - **Posterior** distribution = $P(\theta|y)$
        - **Likelihood** function = $P(y|\theta)$
        - **Prior** distribution = $P(\theta)$
        - **Marginal likelihood** = $P(y)$→ fixed once data is gathered
    - **MLE: maximum likelihood estimate**:
        - Finds the θ that maximizes the **likelihood function**$P(y|\theta)$
- Normal distribution to the posterior distribution
    - (1) approximating the posterior
        - ⇐ works in **low-dimensional parameter space** (theta space)
    - (2) computing the **marginal distributions** for few of the parameters
        - ⇐ works even in **high-dimensional** parameter space
    - (3) intractable **closed-form** analytical solutions
        - ⇐ through MCMC simulations
        - ⇐ when working with complex models
    - Assumptions
        - Independent and identically distributed (i.i.d) data

Mercer, A. (2016, September 8). 5 key things to know about the margin of error in election polls. *Pew Research Center*

This article gives an explanation of survey errors and what they mean. The article describes the common understanding of the margin of error reported in surveys. The margin of error is often reported in polling and other survey data, and it is important to understand the link between the sample size and the sampling error of the survey.

-

Roper Center for Public Opinion Research. (n.d.). *Polling Fundamentals*.

Read the "Total survey error" section of this article. The rest is optional. This section gives a more detailed view of survey errors than the previous reading. Take note of how it differs from the previous reading and identify where this more detailed understanding of the margin of error might differ from the more common understanding in the previous article.
We also use this reading in the pre-class work.

-

(Optional) Lambert, B. (2013, August 28). *Central limit theorems: An introduction* [Video file]

Retrieved from https://www.youtube.com/watch?v=O5wuYAn9Q8c
This is another video deriving and explaining the intuition behind the Central Limit Theorem.

-

Back to Top

(Optional) Osgood, B. (2008, July 3). Central Limit Theorem and Convolution [Video file]. *The Fourier Transforms and its Applications*. Stanford University.

Retrieved from https://www.youtube.com/watch?v=LA4Uv6PMRTM
Focus on the following segments of the video:

- 08:30–12:00: A demonstration on how convolution leads to a normal shaped curve.
- 17:40–19:50: This video talks about convolution, but the CLT is about averaging. This segment shows how convolution and averaging are related and why both lead to a normal curve.
- Everything from 20:00 onwards is less relevant, but for those of you who know about convolutions and would like to understand how these relate to probability distributions in general, watch the whole video.

Class

Back to Top

CS146 6.2 - Multinomial likelihoods with conjugate priors

Pre-class

Gist: https://gist.github.com/AntonioStark/41d1826448f02a366d2276075426485d

Study guide

- **Binomial distribution**
    - # of successes in *n* trials
    - (success,fails) = $(s, f) \sim Binomial(n, (p, 1-p)\ )$
- **Multinomial model**
    - # of times a **category** is observed in *n* trials
    - Has a **parameter/probability vector**
      (expressed as a **simplex**: a vector whose elements are real numbers of [0,1] and sum of all is 1)
    - **stats.multinomial.rvs()**: draw random samples from a multinomial distribution
    - **stats.multinomial.pmf()**: probability mass function of a particular (set of) count vector(s)
- **Dirichlet distribution**
    - **Conjugate prior** to the multinomial likelihood
    - Support: all **length-k** probability vectors
      ⇒ **(k-1) dimensional simplex**
    - **Parameter** of Dirchlet distribution $\vec{\alpha}$: **k-dimensional vector** (does **not** have to be a simplex)
    - **Samples** of the Dirichlet: a **simplex**
    - **Larger sum** of $\vec{\alpha}$: **small variance** (i.e. samples are more concentrated)
    - ⇔ **smaller sum** of $\vec{\alpha}$: **small variance** (i.e. samples are more concentrated)
    - **scipy.dirichlet.rvs()**: generate samples
    - **scipy.dirichlet.pdf()**: evaluate pdf

Readings

[mathematicalmonk]. (2011, June 25). *Dirichlet distribution*

This video gives a nice introduction to the Dirichlet distribution — its algebraic form, how to conceptualize it, and how to visualize it. Statistics like the mean, mode, and variance are also derived. With this video and the next, focus on how to interpret samples from the Dirichlet distribution, and how to interpret the α vector parameter of the distribution.

- $\theta \sim Dir(\alpha)$
    - $\theta = (\theta_1, \ldots, \theta_n)$(i.e. an **n-dimensional vector**)
    - $\alpha = (\alpha_1, \ldots, \alpha_n), \alpha_i > 0$

Back to Top

- $p(\alpha) = \frac{1}{B(\alpha)} \prod_{i=1}^{n} \quad {\theta_i}^{\alpha_i - 1}$
  - Support (range): $\theta \in S$, a **probability simplex**
    - $S = \{x \in R^n : x_i \geq 0, \sum_{i=1}^{n} \quad x_i = 1\}$
  - $\frac{1}{B(\alpha)} = \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1)\ldots\Gamma(\alpha_n)} = \frac{\Gamma(\sum_{i=1}^{n} \quad \alpha_1)}{\prod_{i=1}^{n} \quad \Gamma(\alpha_i)}$(inverse of a Beta function of alpha)
    - $\alpha_0 = \sum_{i=1}^{n} \quad \alpha_i$
- Mean: $E(\theta_i) = \frac{\alpha_i}{\alpha_0}$
- Mode: $\frac{\alpha_i - 1}{\alpha_n - n}$, i.e. when θ is a vector of $(\frac{\alpha_1 - 1}{\alpha_n - n}, \ldots, \frac{\alpha_n - 1}{\alpha_n - n})$
- Variance: $\sigma^2(\theta_i) = \frac{\alpha_i(\alpha_9 - \alpha_i)}{{\alpha_0}^2(\alpha_0 + 1)}$

[mathematicalmonk]. (2011, June 25). *Dirichlet–Categorical model*

The Dirichlet distribution is a **conjugate prior for the multinomial distribution** (here incorrectly referred to as the categorical distribution). The posterior distribution under the multinomial likelihood and Dirichlet prior is derived

- $X_1, \ldots, X_n \sim multinomial(\theta)$
  - ⇔ $P(X_i = j | \theta) = \theta_j$
  - Probability that X$_i$ takes a particular value: $\theta = (\theta_1, \ldots, \theta_n)$
    - $\theta_i \geq 0, \Sigma\theta_i = 1$
  - X only takes a particular value in a set, $X_i \in \{1, \ldots, m\}$
    - ⇒ here, there are m values that X can take
    - ⇔ X is a multinomial of m categories
- Data: $(x_1, \ldots, x_n), x_i \in \{1, \ldots, m\}$

Sections 53 (pg. 525), 62 (pg. 559) Stan Development Team. (2017, December 11). *Stan modeling language: User's guide and reference manual*

Read Sections 53 (p525) and 62 (p559) to see how the Multinomial and Dirichlet distributions are represented in Stan. We use these distributions during a class activity.
Note that the Categorical distribution is also available, but we don't use it in class. The Categorical distribution (analogous to the Bernoulli distribution) models a single sample from a set of categories. The Multinomial distribution (analogous to the Binomial distribution) models a vector of counts of the frequency with which each of the categories was observed. The Dirichlet distribution is a conjugate prior for the Multinomial distribution and the Categorical distribution — just like the Beta distribution is a conjugate prior for the Binomial distribution and the Bernoulli distribution.

- 53. Multivariate discrete distributions
- 53.1 **Multinomial distribution**
  - Probability mass function
  - Sampling statement
    - *y* ~ **multinomial**(*theta*);
      Increment log probability with multinomial_lpmf(y | theta)
  - Stan functions
    - real **multinomial_lpmf**(int[] *y* | vector *theta*);
      Log multinomial probability mass function with outcome array y of size K, given K-simplex distribution parameter *theta* and implicit total count N=sum(y)

Back to Top

        - Int[] **multinomial_rng**(vector *theta*, int *N*);
          Generate a multinomial variate with simplex distribution parameter *theta* and total count *N*. Can only be used in generated quantities block
- 62. Simplex distributions
    - Support: **K-simplex** for a specified K
    - K-dimensional vector θ is a K-simplex if θ_k>=0 for all k ∈{1,...,K}, and $\Sigma_{k=1}^{K}\theta_k$=1
- 62.1 Dirichlet distribution
    - $Dirichlet(\theta|\alpha) = \frac{\Gamma(\sum_{k=1}^{K} \alpha_k)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} \theta_k{}^{\alpha_k - 1}$
      here $K \in N, \alpha \in (R^+)^K, \theta \in K - simplex$
    - Sampling statement
        - *theta* ~ dirichlet(*alpha*);
          Increment log probability with dirichlet_lpdf(*theta* | *alpha*)
    - Stan functions

Betancourt, M. (2014, November 17). *Hamiltonian Monte Carlo and Stan*

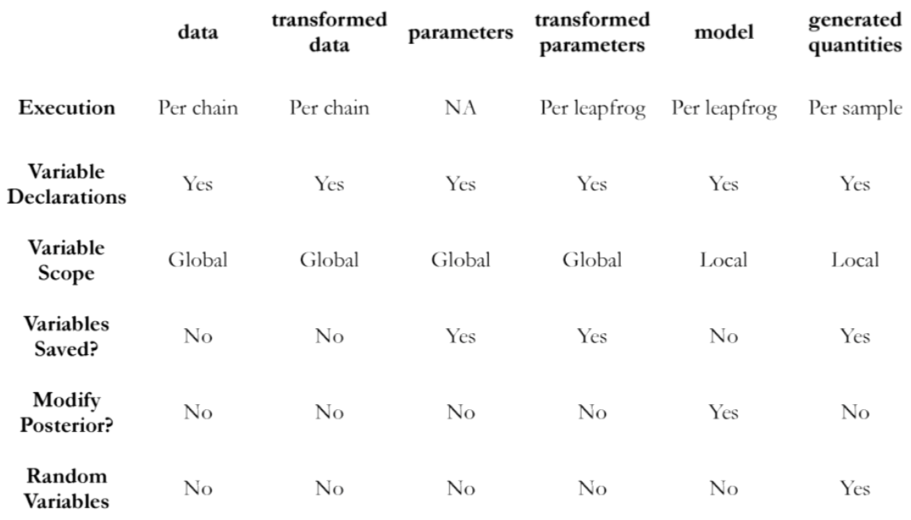
Watch the first 19 minutes of this video. It gives a detailed overview of the Stan modeling language and how to specify a generative statistical model. Pay particular attention to the data, parameters, and model blocks in the Stan code. The other blocks are for more advanced usage and we will get to them later in the course. You should also pay attention to the data types Stan can handle — these are described from about 10 minutes into the video.

You have to be able to read and write Stan models. Make sure you can translate from Stan code to the mathematical representation of a model and vice versa.

The rest of the video is optional, but worth watching if you want to see a real-world model implemented in Stan.

- Programming blocks
    - 1. **Data**: reads external information
        - Read in the local programming languages (Python, R, etc.)
    - 2. **Transformed data**: preprocessing of the data
    - 3. **Parameters** (required)
    - 4. **Transformed parameters**: parameter processing before posterior is computed
        - Do **not** get sampled by the Markov chain
    - 5. **Model**: define the posterior
    - 6. **Generated quantities**: post processing

Back to Top



	data	transformed data	parameters	transformed parameters	model	generated quantities
Execution	Per chain	Per chain	NA	Per leapfrog	Per leapfrog	Per sample
Variable Declarations	Yes	Yes	Yes	Yes	Yes	Yes
Variable Scope	Global	Global	Global	Global	Local	Local
Variables Saved?	No	No	Yes	Yes	No	Yes
Modify Posterior?	No	No	No	No	Yes	No
Random Variables	No	No	No	No	No	Yes

-
- Primitives
  - **Types**: Int / real
  - Stan always samples from an unbounded R^N `⇐ all limits are` **transformed**
  - **Linear algebra types**
    - Column vector: vector[n] name or matrix[n,1] name
    - Row vector: row_vector[n] name or matrix[1,n] name
  - **Arrays**
    - real name[n];
    - vector [n] name [n];
    - matrix[n,n] name [n];
  - **Higher-order objects**
    - simplex[n] theta
    - ordered[n] o; (first element is smallest)
    - positive_ordered[n] p; (last element is smallest)
    - corr_matrix[n] C;
    - cov_matrix[n] Sigma
- Statements
  - if/then/else
  - for(i in 1:I)
  - while (i<I)
- Modifying the posterior
- Sampling statements: **vectorized**
  - `⇒ no need to define per element`

Class

Back to Top

CS146 5.2 - Automated Inference Using Stan

Pre-class

https://gist.github.com/AntonioStark/1ff4da677f511bb20821392cf686d375

Study guide

- Stan
    - Interpreting **standard errors** of the Stan output
    - Analyzing contribution of **priors** to the **posteriors** of the model
    - Reason to check the **model** while modelling

Readings

Watch up to 0:22:30 of Gelman, A. (2016, October 25). *Introduction to Bayesian Data Analysis and Stan with Andrew Gelman*

This video introduces Stan, a package developed by Gelman's group for doing Bayesian inference. We will be using PyStan (the Python interface to Stan) in class. The focus here is to understand how to represent a model in the Stan language. The main example in this first part of the video is a hierarchical model of soccer team strength, used to rank teams based on results from the 2014 FIFA World Cup.
Watch the rest of the video if you have time. It has a really neat geometry-based model for golf later on, and Gelman gives all sorts of practical tips for data modeling through the video.

- Lessons from the world cup example
    - Model the **score differential** not the wins/loses ⇒ this is because you want to model the **process**, not the end data itself
    - Jump in and **fit** the model, and then **check** for its fit
        - No use in staring at the data for infinity trying to figure out the best model. Just go fit it with some model first
    - Combine **sources of information**
    - Compare different **model fits graphically**
        - Models with / without prior: see how the prior information influence the posterior
- Model checking/improvement
    - Does the inference make sense?
        - ⇒ is it coherent with the **prior information** (that's not necessarily in the "prior"
    - Is the model **consistent** with the **data**?
        - ⇒ sometimes when there are two few parameters
    - Expand the model
    - Include more data
- What is Bayes?
    - Data + regularization

Back to Top

Stan Development Team. (2017, December 11). *Stan modeling language: User’s guide and reference manual*

This is the reference manual for Stan. You do not need to read through it before class, but it is useful for looking up how distributions and other functions work in Stan's modeling language.

-

Class

Back to Top

CS146 5.1 - Model Comparison II

Pre-class

Github: https://gist.github.com/AntonioStark/aa6572722bdbfe0707567f18eb27156b

Study guide

- Bayes' Rule w/ models
    - $P(\theta|D,M) = \frac{P(D|\theta,M)\ P(\theta|M)}{P(D|M)}$
    - M (model) is interchangeable with H (hypothesis)
- Probability of a model being the best choice
    - $P(M_i|D) \propto P(D|M_i)P(M_i)$
- **Marginal likelihood** (a.k.a. **Evidence** term)
    - $P(D|M)$
    - Provides link between **model fitting** and **model comparison**
    - **Model fitting** (i.e. inference; i.e. calculating posterior distribution over model parameters):
    $$P(\theta|D,M) = \frac{P(D|\theta,M)\ P(\theta|M)}{P(D|M)}$$
    - **Model comparison**:
    $$P(M|D) = \frac{P(D|M)\,P(M)}{P(D)}$$
    - **Evidence** $= P(data) = \int_{\theta_{min}}^{\theta_{max}} P(data|\theta)\,P(\theta)\,d\theta$
        - $P(data|\theta)$:likelihood
        - $P(\theta)$:prior
- **Bayes factor**: used to compare two models
    - **Ratio** of the marginal likelihood ⇒ prefers the **larger** number model
    - Sensitive to modeling assumptions, esp. The priors
    - Even when priors are **weakly informative**
    - ⇐ **undesirable** when making **predictions**
        - The quality of predictions are useful
        - Test using **cross-validation**
    - ⇐ **useful** when deciding on **hypotheses**
- Numerical calculation of integrals
    - Single variable: **quad**
        - Returns: **estimated value of integral + upper bound on error** (required for counting significant digits)
    - Multiple variables: **nquad**
        - **nquad**(f, [[a1,b1], [a2,b2], [a3,b3]]) = $\int_{a_1}^{b_1} \int_{a_2}^{b_2} \int_{a_3}^{b_3} f(x_1,x_2,x_3)dx_3dx_2dx_1$

Back to Top

Readings

Lambert, B. (2018, May 17). Introducing Bayes factors and marginal likelihoods

This video introduces the marginal likelihood and Bayes factors by building on Bayes equation. The four factors in Bayes equation are the likelihood, the prior, the posterior, and the marginal likelihood. We have been using the first four factors a lot so far in this course. This video explains why the fourth factor is important.

- **Model comparison**
    - P(model$_1$ | data) vs. P(model$_2$ | data)
    - (a.k.a.) P(M$_1$ | data) vs. P(M$_2$ | data)
    - $P(M_1 \mid data) = \frac{P(data \mid M_1)\, P(M_1)}{P(data)}$
        - $P(data|M)$:**marginal likelihood** of a model
        - $P(M)$:model prior
        - $P(data)$:marginal likelihood for both models
            - $= P(data|M_1)P(M_1) + P(data|M_2)P(M_2)$
    - Calculating **marginal likelihood**: take from Bayes' rule
        - $P(\theta \mid data, M) = \frac{P(data \mid \theta, M)\, P(\theta|M)}{P(data|M)}$
        - Where $P(data \mid M) = \int \quad P(data|\theta, M_1)\; P(\theta|M_1)\; d\theta$ (i.e. marginalizing out the parameter(s) θ
    - Ratio of the probabilities of two models on data (odds):
        - $\frac{P(M_1 \mid data)}{P(M_2 \mid data)} = \frac{P(data \mid M_1)}{P(data \mid M_2)} \cdot \frac{P(M_1)}{P(M_2)}$
            - **Bayes factor**: first term: **ratio of marginal likelihoods**
            - Second term: ratio of **priors**
- Issues
    - Calculating **marginal likelihoods**: inherently difficult to calculate
    - Marginal likelihoods are **sensitive** to **parameter priors** (i.e. $P(\theta|M)$ than P(M)s)
    - Difficulty in estimating **priors** for **models** (simpler models have higher prior probabilities, but by how much?)
    - Difficult to define the **threshold** of preference between models
        - Finding **predictive accuracy** on **out-of-sample**-data is better
            - WAIC: watanabe-akaike information criterion
            - LOO-CV: leave-one-out cross-validation

Lambert, B. (2018, May 15). On the sensitivity of the marginal likelihood to prior choice

This video follows on the previous one in discussing some of the problems with using marginal likelihoods and Bayes factors to compare models. Particularly in the case where we want to use a model for making predictions, which happens a lot, other goodness-of-fit measures might be better than Bayes factors. You will learn about one such alternative measures (posterior predictive checks) in a future session.

-

Back to Top

Chapter 28 up to the end of 28.2 of MacKay, D. J. C. (2003). *Information theory, inference, and learning algorithms*

This chapter motivates for Bayesian model comparison as codifying Occam's razor ("among competing hypotheses, the one with the fewest assumptions should be selected" — Wikipedia) within the framework of probability theory. Focus on how Bayes' theorem is used to determine how much the data favor one model (or hypothesis) over another.

- **Occam's razor**
    - Accept the simplest explanation that fits the data
- Model comparison
    - **Plausibility of models given data**
        - $\frac{P(H_1|D)}{P(H_2|D)} = \frac{P(H_1)P(D|H_1)}{P(H_2)P(D|H_2)}$
            - $P(H_1)/P(H_2)$:initial beliefs towards hypotheses
            - $P(D|H_1)/P(D|H_2)$:how well observed data predicts hypotheses
                - ⇒ **simpler** models make **precise** predictions
                - ⇐ **complex** models make a **greater variety** of predictions
- **Data modeling**'s two levels of inference
    - #1: **model fitting**: inference of values of free parameters given model
        - Assumption that model is true
        - Outcome: what are the most probable parameter values
        - **Posterior probability** of parameter(s):
          $$P(\theta|D, H_i) = \frac{P(D|\theta, H_i)\, P(\theta|H_i)}{P(D|H_i)}$$
        - Represented as θ_{MP}(**most likely parameter(s)**) and **error bars**
        - θ_{MP} usually calculated through **gradient**-based methods
    - #2: **model comparison**
        - Posterior probability of **model**
        - $P(H_i|D) \propto P(D|H_i)\, P(H_i)$

Sections 3.2–3.4 of MacKay, D. J. C. (2003). *Information theory, inference, and learning algorithms*

These sections contain two detailed model comparison example. Focus on how the evidence terms of the two models, H_0 and H_1, are compared to determine how much more probable the one model is than the other, given the data. There is an important difference between the first example (sequences of a and b) and the second example (legal evidence). In the first example, we probably want to be able to predict future a-b sequences — this is exactly the scenario that Lambert criticizes in the videos above. In the second example, there is no prediction involved; only whether one hypothesis about a crime is more probable than another hypothesis. Be sure to understand how these two types of scenarios — predictive vs historical — are different.

-

Back to Top

Nguyen, A. (2018). *Integral calculus: Probability edition*

This tutorial contains everything you need to know about integral calculus for working with probability theory. It is worth reviewing everything in the tutorial. For today's session focus on the example in Section 4, which shows how we can look up the definite integral of a pdf since we know what its normalization constant is supposed to be. You will be asked to do an integral of a pdf in the preparatory assessment poll.

-

The SciPy community. (2018, May 5). Integration (scipy.integrate). *SciPy Tutoria*

This tutorial shows you how to use the quad functions in SciPy to do numerical integration. When we cannot solve a definite integral analytically, numerical integration is another option. There will always be some numerical error in your results. Read the sections titled "General integration (quad)" and "General multiple integration (dblquad, tplquad, nquad)". You will need to use the "nquad" function in your pre-class work (or the "dblquad" function if you prefer).

-

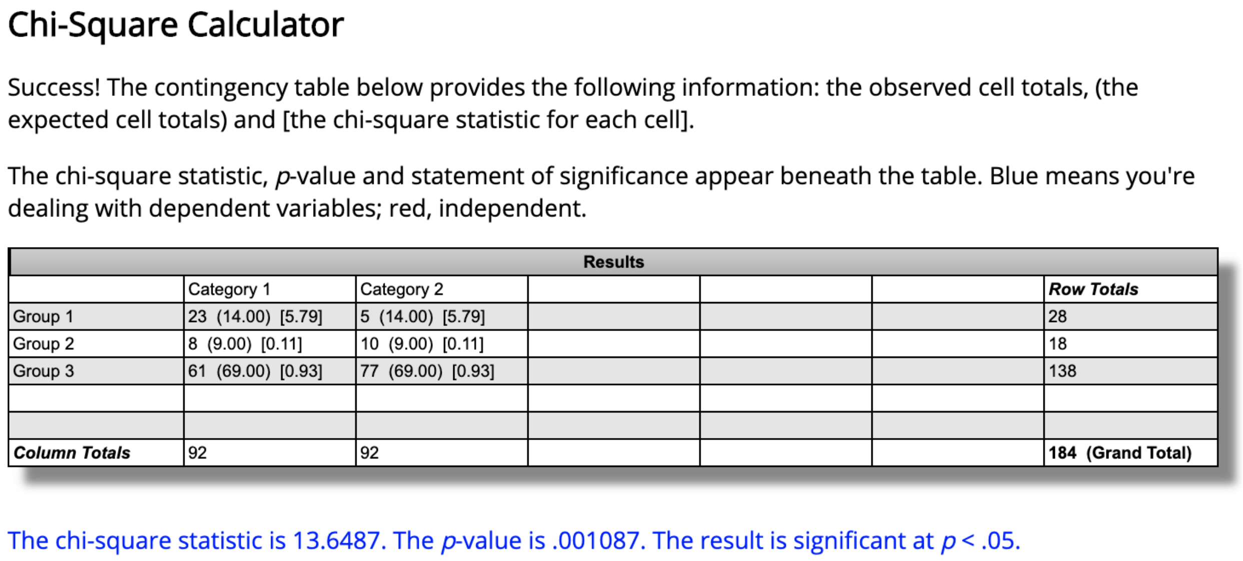
Class

Back to Top

CS146 4.2 - Model Comparison I

Pre-class

(I also just built my own calculator out of Google Sheets:)
https://docs.google.com/spreadsheets/d/13PHDiiMBrPT9EEeIfzBTa3AgkFWfIt7n_E9cIqoD1uI/edit?usp=sharing
For some reason, the paper divides by 185 instead of 184, giving different Expected and Chi-squared values
P-value is 0.0011, so p<0.05 and p<0.005. Our data is statistically significant.



Chi-Square Calculator

Success! The contingency table below provides the following information: the observed cell totals, (the expected cell totals) and [the chi-square statistic for each cell].

The chi-square statistic, *p*-value and statement of significance appear beneath the table. Blue means you're dealing with dependent variables; red, independent.

Results

	Category 1	Category 2				*Row Totals*
Group 1	23 (14.00) [5.79]	5 (14.00) [5.79]				28
Group 2	8 (9.00) [0.11]	10 (9.00) [0.11]				18
Group 3	61 (69.00) [0.93]	77 (69.00) [0.93]				138
Column Totals	92	92				**184 (Grand Total)**

The chi-square statistic is 13.6487. The *p*-value is .001087. The result is significant at *p* < .05.

Study guide

- Comparing models & hypotheses
  - **Frequentist** significance test vs. **Bayesian** posterior probabilities
  - Computing **p-values**
  - Communicating **probability** & **risk**
- **P-value**:
  - Need a **test statistic** under the **null hypothesis** assumption
  - ⇒ a **distribution** b/c of random variation from the null hypothesis
  - ⇒ about **null hypothesis** but not about **model**
  - ⇒ does **not** say anything about the true/false of the **null hypothesis**
  - ⇒ only about the **probability** of observing the **data** under the null hypothesis
- **Chi-squared** test
  - Statistical significance on **categorical** data
  - ⇒ build a **contingency** table
  - **Null** hypothesis: **all groups** follow the same distribution (i.e. both treated & placebo patients have same results)

Back to Top

Readings

Resnick, B. (2017, July 31). What a nerdy debate about p-values shows about science — and how to fix it. *Vox*

There is currently an important debate around the usefulness and common misuses of p-values in scientific publications. This article gives a summary of why looking at p-values to establish statistical significance is often problematic in practice. You should focus on understanding precisely what a p-value means – that is, what is the correct interpretation of a sentence like "we achieve statistical significance at a p-value < 0.01". Read these two sections carefully–"What is a p-value?" and "You might be thinking: Isn't this a pretty roundabout way to prove an experiment worked?".

- **Replication crisis**: experiment results cannot be replicated
- **P-value** of 0.05 → 0.005
    - **P-hacking** and **outcome switching** is making it easy to reach 0.05
- Disproving a **null hypothesis**: eliminating an alternate explanation for the results
    - "Innocent until proven guilty" - the Regina Nuzzo of science
- **P-value**: how **rare** is the result **<u>if null hypothesis = True</u>**?
    - ≠ p-value of experiment result being random chance - which **already** makes a statement about the state of the world w/ or w/o the null hypothesis (i.e. **false positive**)
- The **false positive** rate can be much higher than the p-value
- If the **experimental hypothesis** is true, the p-value should be **much lower**

McHugh, M. L. (2013). The chi-square test of independence. Biochemia Medica, 23(2), 143–149

This is a tutorial on the chi-square test statistic. Focus on the assumptions of the test, working through the case study, and the interpretation of chi-square values. This article is also used as part of the pre-class work where you reproduce the results in the Case Study section.

- **Chi-square test of independence** (aka Pearson Chi-square test)
    - Significance for observation
    - + which categories **account** for differences found
- **Non-parametric** statistics (i.e. distribution-free test)
    - Level of measurement is **nominal/ordinal**
    - **Sample sizes** of study groups are unequal
- Chi-square assumptions
    - **Data**: frequencies or count of cases
    - Levels/categories: mutually exclusive
    - Each subject contribute only to one cell in X^2
    - Count of **expected** should be >= 5 in at least 80% of the cells
        - No cell should have # of expected < 1
        - ⇒ met when sample size >= 5*number of cells
- Calculating Chi-square
    - $\chi^2 = \Sigma \quad \chi_{ij} = \Sigma \quad \frac{(O_{ij}-E_{ij})^2}{E_{ij}}$

Back to Top

    - **Marginals**: the sum per row or column
    - Expected value of each cell $E_{ij} = \frac{M_i * M_j}{n}$
- Significance level
    - **Degree of freedom (dof)**: (# of rows - 1) x (# of columns -1)
- Per-cell Chi-squared value
    - If <1.0, # of observed cases ≒ # of expected cases
- **Fisher's exact test**
    - Only for 2x2 tables
- **Maximum likelihood ratio Chi-squared test**
    - Use only when sample size is too small
- **Strength test**
    - Statistical significance ≠ clinical importance
- **Cramer's V test**
    - Strength test
    - Form of **correlation**

Spiegelhalter, D. (2013, May 21). Communicating risk and uncertainty

Computing risk and presenting numerical results to a trained statistician is very different from communicating results to an average person. This is especially important in medical research and practice, where doctors have a duty to help their patients understand the risks of various diseases and treatments so they can make informed decisions. Watch the whole video and focus on which types of communication are effective based on the available evidence.

-

Chapter 6 of Gelman, A., et al. (2013). Bayesian data analysis, third edition. Boca Raton, Fla: Chapman & Hall

(Optional) This chapter is on the use of test statistics, normally associate with frequentist statistics, with Bayesian data modeling. This is challenging material, but read it to develop an understanding of using test statistics commonly used for calculating p-values, to identify shortcomings in your statistical model. Well-designed test statistics can help you extend your model in a way that better explains your data set. This is one of the ways of deciding which probability distributions to use — determine whether the chosen probability distribution follows the same distribution over a test statistic as the data set does.

-

Class

Back to Top

CS146 4.1 - Selecting prior hyperparameters

Pre-class

Github: https://gist.github.com/AntonioStark/ed374a1eb8f51c8d0b53b5754b651178
Optimization method:
started at: (μ,v,α,β)=(0.00, 1.00, 3.00, 3.00)
optimization successful after 34 iterations,
ending function value is 1.28e-08
ended at: (μ,v,α,β)=(2.30, 11.00, 9.56, 23.55)
Optimization validation:
for 100 samples from the prior of (μ,v,α,β)=(2.30,11.00,9.56,23.55):
data mean is: 2.30 ± 0.50
estimated mean is: 2.33 ± 0.43
data variance is: 2.75 ± 1.00
estimated variance is: 2.63 ± 0.88
Analytical method:
solved for: (μ,v,α,β)=(2.30, 11.00, 9.56, 23.55)

Study guide

- **Normal** vs. **log-normal** distribution
    - **Normal** distribution
        - For quantities of **sums** of independent random variables
        - **Support**: real numbers (R)
        - Distribution: **symmetric**
        - **Additive** effects among variables
    - **Log-normal** distribution
        - For quantities of **product** of independent random variables
        - If **x** follows a log-normal, **log x** is normally distributed
        - **Support**: **positive** real numbers (R^+)
        - Distribution: **non-symmetric,** skewed to the right
        - **Multiplicative** effects among variables
    - Normal <-> log-normal
        - The **poster** parameters mean μ and variance σ^2 in normal
        - = equal to **posterior** parameters mean μ and variance σ^2 in log-normal
- Normal-inverse-gamma **NIG** distribution
    - Prior hypermarameters: $\mu_0, \nu, \alpha, \beta$

Back to Top

Readings

Scheffler, C. (2018). Optimization in SciPy

These notes explain how to use the minimize() function in SciPy for finding a minimum (or maximum) of a function. Focus on actually running the code provided in the examples. Compare the output from your code to what you would expect the solution to each problem to be, based on the information in each example. Experiment with different functions or different initial values. You should know how to maximize or minimize a function of 1 or more variables to be ready for class.

-

Limpert, E., Stahel, W.A., Abbt, M. (2001, May). Log-normal distributions across the sciences: Keys and clues. *BioScience*

This is background reading on the motivation for using the log-normal distribution to model many natural phenomena. Focus on understanding the distinction between additive effects (often modeled using the normal distribution) and multiplicative effects (better modeled using the log-normal distribution) and the arguments for why many natural phenomena arise from multiplicative effects.

- **Log-normal** distributions are useful when:
    - **Low mean** values
    - **Large variances**
    - **Non-negative** values
    - ⇒ **skewed** (to the right)
- Examples
    - Biological mechanisms of **exponential growth** with **symmetrical variation**
- **Central Limit Theorem (CLT)**
    - When there is an **infinite** number of independent, identically distributed random variables:
    - The **sum** of variables approach a **normal** distribution
    - The **product** of variables approach a **log-normal** distribution
- **Log-normal** distribution:
    - The **product** of independent log-normal quantities is also log-normal
    - Median of product = product of median of the factors
    - In mathematics (wikipedia):

$$p(x \mid \mu, \sigma) = \frac{1}{x\sigma\sqrt{2\pi}} exp\ (-\frac{ln^2\ (x-\mu)}{2\sigma^2})$$

    - In Scipy:

$$f(x,s) = \frac{1}{sx\sqrt{2\pi}} exp\ (\frac{ln^2\ (s)}{2s^2})$$

        - Equivalent when s=sigma, scale=exp(mu)
    -

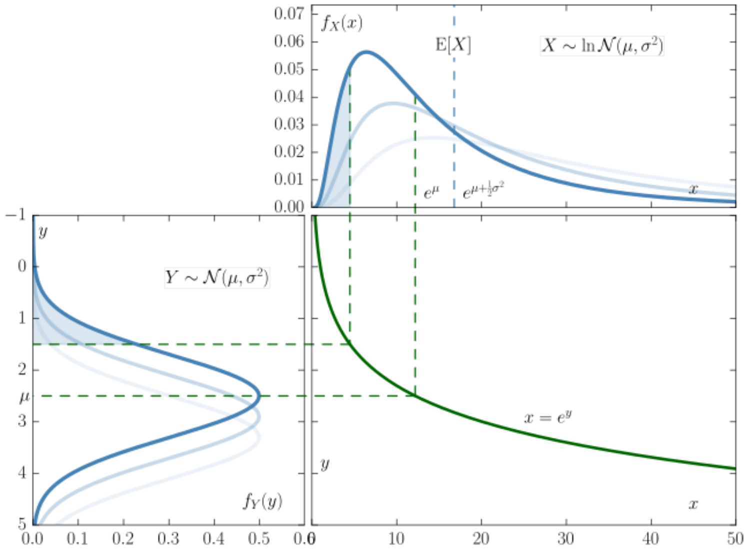
Log-normal distribution. (n.d.). In *Wikipedia*.

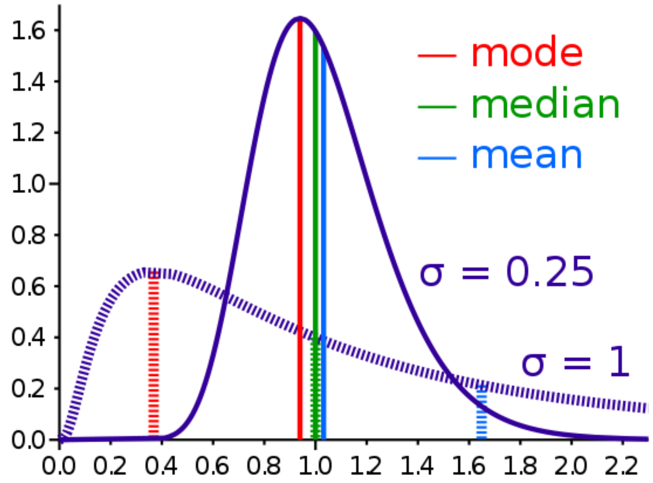
Reference for the log-normal distribution on Wikipedia. Use this to look up mean, variance and any other values of interest of the distribution.

- Generation and parameters

Back to Top

- When *z* is a standard normal variable
- $X = e^{\mu+\sigma z}$is a normal distribution with parameters μ, σ
- **Geometric (multiplicative) parameters**
  - μ^* = e^{μ}: the **median** of the distribution
  - σ^* = e^{σ}

- 

- 

- Relationships of $X \sim LogNormal(\mu, \sigma^2)$
  - $aX \sim LogNormal(\mu + \ln a, \sigma^2)$
  - $1/X \sim LogNormal(-\mu, \sigma^2)$
  - $X^a \sim LogNormal(a\mu, a^2\sigma^2)$
- Multiplication/division between independent, log-normal random variables X_1, X_2
  - $\mu = \mu_1 + \mu_2$or $\mu = \mu_1 - \mu_2$
  - $\sigma^2 = {\sigma_1}^2 + {\sigma_2}^2$
  - Generalization: for $X_j \sim LogNormal(\mu_j, {\sigma_j}^2)$

$$Y = \prod_{j=1}^{n} X_j \sim LogNormal(\sum_{j=1}^{n} \mu_j, \sum_{j=1}^{n} {\sigma_j}^2)$$

- **Multiplicative Central Limit Theorem (M-CLT)**
  - The geometric mean of *n* independent, identically distributed, positive random variable X_i

Back to Top

- As $n \to \infty$,
- $\mu = E[\ ln\,(X_i)\ ]$
- $\sigma^2 = var[\,ln\,(X_i\ )]\ /\ n$

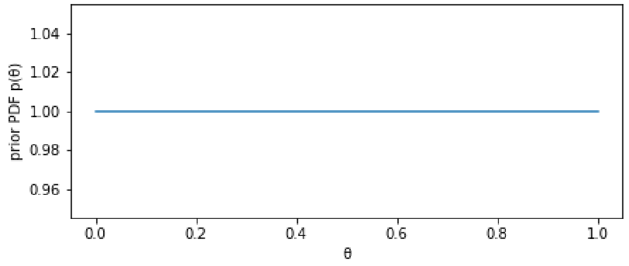
Class

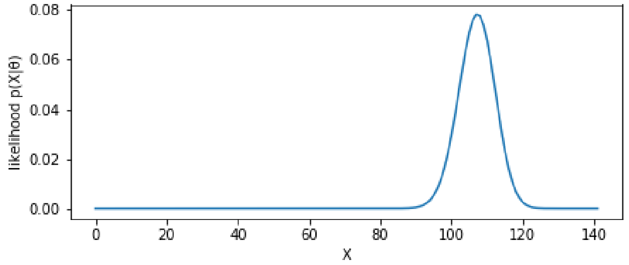
Back to Top

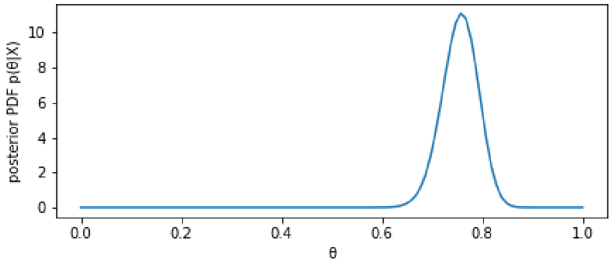
CS146 3.2 - Making predictions and decisions

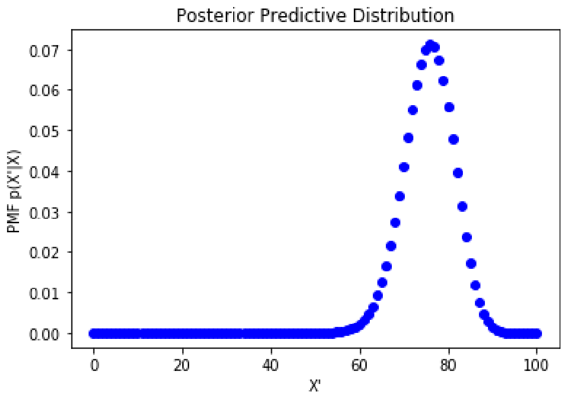
Pre-class

Github: https://gist.github.com/AntonioStark/62d7675b830b2fdfb6d3f2cf65650e90









Study guide

Readings

Lambert, B. (2014, August 12). *Bayesian inference in practice — Posterior distribution example: Disease prevalence*

Introduction to using a binomial likelihood and beta conjugate prior for modeling disease data. This example gets used in all other videos below, so make sure you understand how it works. We look at a similar scenario (medical trial data) in the pre-class work.
Note that he uses the uniform distribution as a prior and that the uniform distribution on the interval [0,1] is exactly the same as the Beta(1,1) distribution. We use this fact in class too.

Back to Top

Also, note that this is another example of a conjugate prior distribution as we had in the previous lesson. That is why the posterior distribution is so easy to derive from the prior and the data.

- Problem description
  - θ: % of disease in a **population**
  - Sample data
  - **Likelihood**: p(data | θ) = **binomial** distribution
  - **Prior**: p(θ) = **beta** distribution (1,1) = uniform distribution over range [0,1]
  - **Posterior**: p(θ | data) = **beta** b/c beta is **conjugate prior** to **binomial**
- **Beta - binomial - beta**
  - Prior: Beta(α, β) (here α=1, β=1)
    - Interpretation: (α-1) successes, (β-1) failures
  - Posterior: Beta $(\alpha + \sum_{i=1}^{n} \quad x_i, \beta + n - \sum_{i=1}^{n} \quad x_i)$
    - A.k.a. Beta(α+X, β+N-X)
  - Beta distribution:
    - $E[\theta] = \frac{a}{a+b}$

Lambert, B. (2014, August 12). Bayesian inference in practice — Disease prevalence

This video shows how the posterior mean is related to the prior mean and maximum likelihood estimate for the mean from the data.

- For a beta distribution of **beta(a,b)**,
  - (prior) $p(\theta) \sim beta\ (a, b)$
  - (prior) $E[\theta] = \frac{a}{a+b}$
  - (posterior) $p(\theta|X)\ \sim beta(a + X, b + N - X)$
  - (posterior) $E[\theta|X] = \frac{a+X}{a+b+N}$
- Posterior mean:
  - $E[\theta|X] = \frac{a+X}{a+b+N}$
    $=\frac{a}{a+b}\frac{a+b}{a+b+N} + \frac{X}{N}\frac{N}{a+b+N}$
  - $\therefore \mu_{posterior} = \mu_{prior} w(a, b, N) + \mu_{ML} w'(a, b, N)$
    - $\because$ prior mean: $\mu_{prior} = \frac{a}{a+b}$
    - $\because$ maximum likelihood (ML) mean: $\mu_{ML} = \frac{X}{N}$
    - $w(a, b, N) = \frac{a+b}{a+b+N}$(w↑ : a↑, b↑, N↓): weight to prior
    - $w'(a, b, N) = \frac{N}{a+b+N}$(w↑ : a↓, b↓, N↑): weight to likelihood

Lambert, B. (2014, August 12). Prior and posterior predictive distributions — An introduction

This video explains the main concepts behind predictive distributions. It is important to note that a predictive distribution is the expected value of the likelihood function under the prior or the posterior over the parameters. This gives us either the prior predictive distribution (what we think the data will look like before we have observed the data) or the posterior predictive distribution (what we think future data will

Back to Top

look like after we have already observed some data, and calculated a posterior over our parameters). Working with predictive distributions is the main focus of this session.

- **Prior predictive distribution**: expectation of parameters **before** we see the data
  - = **marginal probability** of the data
  - $p(y) = \int_{\theta\in\Theta} \quad p(y,\theta)\, d\theta$

    $= \int_{\theta=\Theta} \quad p(y|\theta)\, p(\theta)\, d\theta$

    = ∫ (likelihood) * (**prior**)
- **Posterior predictive distribution**: expectation of parameters **after** we see the data
  - = also a **marginal probability**
  - $p(y' \mid y) = \int_{\theta\in\Theta} \quad p(y',\theta \mid y)\, d\theta$

    $= \int_{\theta=\Theta} \quad p(y' \mid \theta, y)\, p(\theta \mid y)\, d\theta$

    = ∫ (likelihood) * (**posterior**)

    ∵ experiments are **independent** of each other

    ⇔ $p(y' \mid \theta, y) = p(y' \mid \theta)$

    ∵ y' (new data) is independent of y (old data)

Lambert, B. (2014, August 12). Prior predictive distribution: example disease - 1, 2

This shows the derivation of the prior predictive distribution for the disease example. The resulting predictive distribution is a beta-binomial distribution: https://en.wikipedia.org/wiki/Beta-binomial_distribution

You don't need to be able to do the mathematical derivation of the predictive distribution but you should understand the concepts behind predictive distributions.

- **Prior** distribution: continuous **beta** distribution (PDF)
- **Prior predictive** distribution: **discrete** distribution (PMF)
  - $p(X) = \int_{\theta=0}^{1} \quad p(X,\theta)\, d\theta$ (where $X = \sum_{i=1}^{n} \quad x_i$)

$$= \int_{\theta=0}^{1} \quad p(X|\theta)\, p(\theta)\, d\theta$$

= ∫ (likelihood: binomial) * (**prior**: beta)

$$= \int_{\theta=0}^{1} \quad \frac{N}{X}\theta^{X}(1-\theta)^{N-X} \cdot \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\theta^{a-1}(1-\theta)^{b-1}\, d\theta$$

$$= \frac{\Gamma(N+1)\,\Gamma(a+b)}{\Gamma(X+1)\Gamma(N-X+1)\,\Gamma(a)\Gamma(b)}\int_{\theta=0}^{1} \quad \theta^{X+a-1}(1-\theta)^{N-X+b-1}d\theta$$

$\because \frac{N}{X} = \frac{N!}{X!\,(N-X)!}$ & $\Gamma(\alpha+1) = (\alpha)!\ if\ \alpha \in N$

$$= \frac{\Gamma(N+1)\,\Gamma(a+b)}{\Gamma(X+1)\Gamma(N-X+1)\,\Gamma(a)\Gamma(b)} \cdot \frac{\Gamma(a')\Gamma(b')}{\Gamma(a'+b')}\int_{\theta=0}^{1} \quad \frac{\Gamma(a'+b')}{\Gamma(a')\Gamma(b')}\theta^{a'-1}(1-\theta)^{b'-1}d\theta$$

$$when\ a' = X+a, b' = N+b-X$$

$$= \frac{\Gamma(N+1)\,\Gamma(a+b)\Gamma(X+a)\Gamma(N+b-X)}{\Gamma(X+1)\Gamma(N-X+1)\,\Gamma(a)\Gamma(b)\Gamma(a+b+N)} \cdot \int_{\theta=0}^{1} \quad beta(a',b')d\theta$$

$= \frac{\Gamma(N+1)\,\Gamma(a+b)\Gamma(X+a)\Gamma(N+b-X)}{\Gamma(X+1)\Gamma(N-X+1)\,\Gamma(a)\Gamma(b)\Gamma(a+b+N)}$ ⇐ **beta-binomial** distribution

Back to Top

- When a = b = 1:
  - $\Gamma(a+b) = \Gamma(2) = 1! = 1$
  - $\Gamma(a) = \Gamma(b) = \Gamma(1) = 0! = 1$
  - $\Gamma(X+a) = \Gamma(X+1)$
  - $\Gamma(N+b-X) = \Gamma(N-X+1)$
  - $\therefore \frac{\Gamma(N+1)\,\Gamma(a+b)\Gamma(X+a)\Gamma(N+b-X)}{\Gamma(X+1)\Gamma(N-X+1)\,\Gamma(a)\Gamma(b)\Gamma(a+b+N)} = \frac{\Gamma(N+1)}{\Gamma(N+2)} = \frac{N!}{(N+1)!} = \frac{1}{N+1}$
  - = **uniform discrete distribution** of 0 to N

- **Beta-binomial** distribution
  - **Discrete** probability distribution
  - Support: **non-negative integers** (k ∈ {0, … , n})
  - When the probability of success for each **Bernoulli trial**: unknown or random
  - **Conjugate** to the **binomial**
  - Parameters:
    - k ∈ n (# of trials)
    - α, β = R > 0 (positive real numbers)
  - **PMF:**
    - $f(k \mid n, \alpha, \beta) = \frac{n}{k}\frac{B(k+\alpha, n-k+\beta)}{B(\alpha,\beta)}$
    - $= \frac{\Gamma(n+1)}{\Gamma(k+1)\,\Gamma(n-k+1)} \cdot \frac{\Gamma(k+\alpha)\,\Gamma(n-k+\beta)}{\Gamma(n+\alpha+\beta)} \cdot \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}$
  - Approximates the **binomial distribution** when α, β are large

Lambert, B. (2014, August 12). Posterior predictive distribution: example disease

Derivation and demonstration of the *posterior* predictive distribution (you have already seen the prior video above) for the disease example.

This is important for completing your pre-class work. Again, remember that the posterior predictive distribution is the expected value of the likelihood function under the posterior over the parameters of the model. Be ready to calculate more expected values in class.

- **Posterior predictive** distribution:
  - when **prior** is a **beta** distribution $p(\theta) = beta(a, b)$
  - $p(X'|X) = \int_{\theta=0}^{1} \quad p(X'\,|\theta)\,p(\theta\,|X)\,d\theta$
    - ∵ X' (new data) and X (old data) are independent
    - ⇔ **likelihood**: **binomial** distribution $p(X' \mid X, \theta) = p(X' \mid \theta)$
    - **Posterior**: **beta** distribution $p(\theta|X) = beta(X+a, N+b-X) = beta(a', b')$ (new hyperparameters $a' = X+a, b' = N+b-X$)

      $$= beta - binomial(N', a', b')$$

      $$= beta - binomial(N', X+a, N+b-X)$$

- Probability of a **single new data** having a disease:
  - $p(X'=1|X) = \int_{\theta=0}^{1} \quad \theta \cdot p(\theta|X)\,d\theta = E[\theta|X]$

    $$= \frac{a'}{a'+b'} = \frac{X+a}{a+b+N}$$

    ∵ $p(\theta|X)$ is the **posterior** distribution of $beta(a', b')$

    $= \frac{X+1}{N+2}$ when (a=b=1)

Back to Top

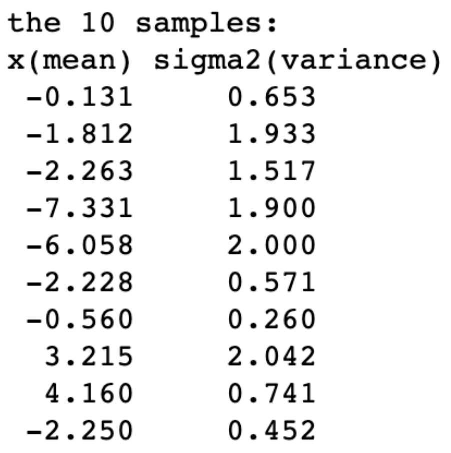
Class

Back to Top

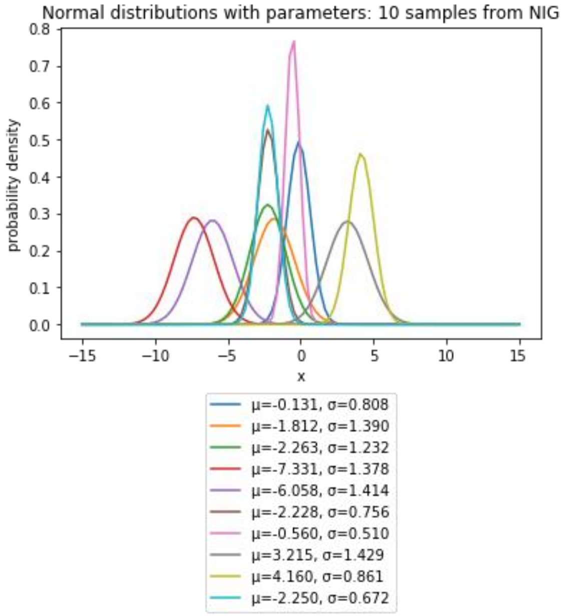
CS146 3.1 - Normal likelihoods with conjugate priors

Pre-class

Github: https://gist.github.com/AntonioStark/acc181663eaa0ad6d7276d091935365b



```
the 10 samples:
x(mean) sigma2(variance)
 -0.131     0.653
 -1.812     1.933
 -2.263     1.517
 -7.331     1.900
 -6.058     2.000
 -2.228     0.571
 -0.560     0.260
  3.215     2.042
  4.160     0.741
 -2.250     0.452
```



Study guide

- Normal distribution
  - **Inference** of mean & distribution vs. **computing** sample mean & distribution
- Normal distribution w/ unknown parameters
  - **Data** (population) ~ N(μ, σ) with unknown μ, σ
  - ⇒ data: real numbers ⇒ **likelihood function**: normal distribution of N(data | μ, σ)
  - Prior: 2-dimensional (μ, σ)
  - Normal-inverse-gamma: **conjugate prior** for normal likelihood
- Parameterizations of the normal likelihood
  -

Parameters of the normal **likelihood**	**Conjugate prior**
Mean (μ) & variance (σ^2)	normal-inverse-gamma
Mean (μ) & standard deviation (σ)	normal-gamma
Mean (μ) & precision (τ)	normal-gamma

  - Variance = (standard deviation)^2 = 1/precision
  - (σ^2) = (σ)^2 = 1/τ
- Conjugate priors

Back to Top

Likelihood function	Conjugate prior (hence posterior)
Exponential distribution	Gamma distribution
Normal distribution, unknown mean	Normal distribution
Normal distribution, unknown variance	Inverse-gamma distribution

    - Why use:
        - Simple and fast **update equations** for getting from prior → posterior distribution
        - Simply calculate **posterior parameters** from **prior parameters + data**
        - We can calculate **expected values** (E[x]) and **integrals** explicitly
            - w/o using **numerical integration** (uses **trapezoid**)
- Parameters
    - **Hyperparameters**: parameters of the **prior** distribution (p(θ))
    - **Parameters**: parameters of the **likelihood** function (p(data|θ))
        - ⇐ unknown variables for which we compute the **posterior** distribution

Readings

Lambert, B. (2018, May 16). *What is a conjugate prior?*

Retrieved from https://www.youtube.com/watch?v=aPNrhR0dFi8&t=6s
Watch this video to get an overview of what conjugate priors are, of why they make Bayesian inference much easier, and to see an example. We start using conjugate distribution in this session and will build on them during the remainder of this unit before getting back to approximate inference methods

- **Conjugate prior**:
    - When a distribution is a **conjugate prior,**
    - when the distribution is a **prior**
    - To a particular **likelihood distribution**,
    - The **posterior** is the same distribution form as the **prior**
- Benefits
    - We don't need to do a math of trying to figure out the form of the posterior may be
    - We can generate the **posterior exactly**
    - Can also provide a list of **realistic** distributions
- Harms
    - The form of prior may be **unlikely** given the data

Chapter 24 of MacKay, D. J. C. (2003). Information theory, inference, and learning algorithms

MacKay describes the process behind inferring the mean and standard deviation of a Gaussian distribution. In class we work with the variance rather than the standard deviation, so the math is slightly different but the same concepts apply in both cases. The mathematical derivations in this chapter are

Back to Top

optional and you don't need to know about "improper priors" or "noninformative priors" for class – just think of these as prior distributions.
Pay particular attention to the discussion around the difference between Frequentist estimates of the mean and standard deviation, and the Bayesian approach of computing a posterior over the mean and standard deviation. This discussion starts from the section titled "Maximum likelihood and marginalization".
Also, pay attention to Figure 24.1 on page 321. Being able to visualize your priors and posteriors is an important skill that we develop in this session.

- **Exact marginalization** (a.k.a. **Integrating** over **nuisance parameters**)
    - 1. Integration (when nuisance parameters are continuous)
    - 2. Summation (when discrete variables)
- 24.1 Inferring the mean and variance of a Gaussian distribution
    - For μ:
        - Hyperparameters μ_0, σ_0 : parameters for **prior** on μ
        - P(μ | μ_0, σ_μ) = Normal(μ ; μ_0, σ_μ^2)
        - Limit: μ_0 → 0, σ_μ → ∞: **noninformative prior** (i.e. flat prior)
            - I.e. **invariant** under $\mu' = \mu + c \Leftrightarrow P(\mu) = \text{constant}$
    - For σ:
        - Gamma distribution
        - Hyperparameters b_β , c_β: parameters for **prior** on σ
        - P(β) = Γ(β ; b_β , c_β) when $\beta = 1/\sigma^2$
        - Mean = $b_\beta c_\beta$
        - Variance = $b_\beta^2 c_\beta$
        - Limit $b_\beta c_\beta = 1$, c_β → 0: **noninformative prior**
            - I.e. **invariant** under $\sigma' = c\sigma$
    - Maximum likelihood & marginalization
        - Given data D=$\{x_n\}_{n=1}^N$,
        - Estimator of μ: $\underline{x} \equiv \sum_{n=1}^{N} \quad x_n/N$
        - Estimator of σ:
            - $\sigma_N \equiv \sqrt{\frac{\sum_{n=1}^{N} \quad (x-\underline{x})^2}{N}}$
            - $\sigma_{N-1} \equiv \sqrt{\frac{\sum_{n=1}^{N} \quad (x-\underline{x})^2}{N-1}}$

Flectcher, T. (2018). Gaussian priors

A derivation of why the inverse-gamma distribution has the appropriate algebraic form to be a conjugate prior for the variance parameter in a Gaussian likelihood function. This should deepen your understanding of the relationship between a likelihood function and its conjugate prior — that they have the same algebraic form with respect to the unknown parameter(s).

-

Back to Top

Murphy, K.P. (2007). Conjugate Bayesian analysis of the Gaussian distribution

If you want the full mathematical treatment of deriving conjugate priors for Gaussian likelihoods with unknown mean and variance, or standard deviation, or precision, work through this paper.

-

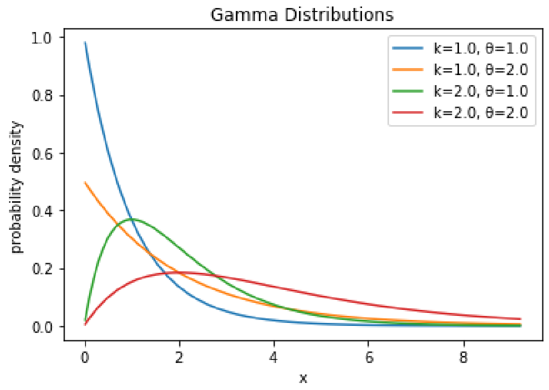
Class

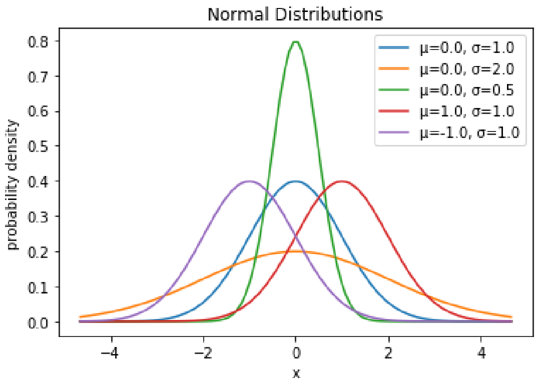
[Back to Top]

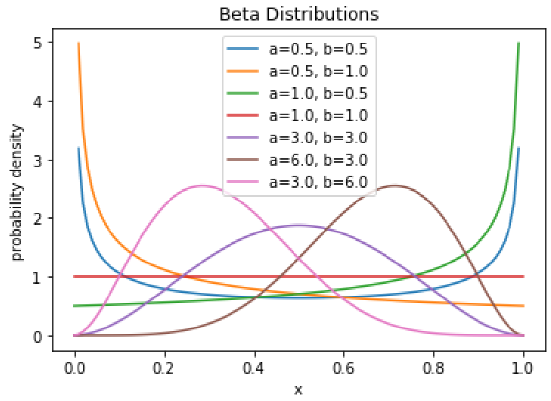
CS146 2.2 - Communicating Results

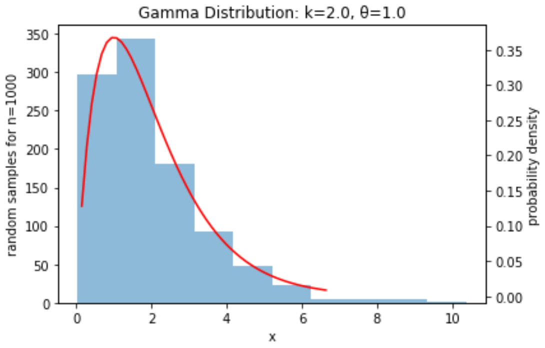
Pre-class

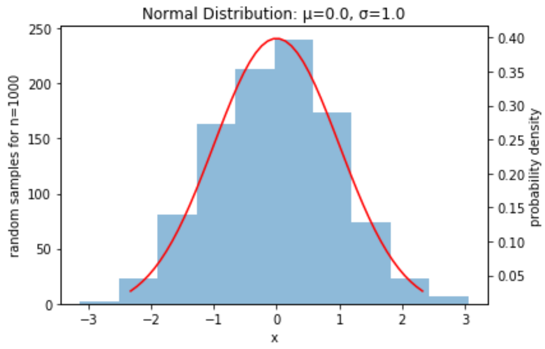
https://gist.github.com/AntonioStark/859e0afe548dc6f5a0fea391bcdced0c

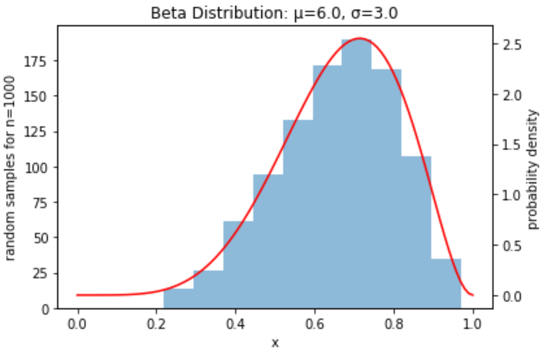

Gamma Distributions


Normal Distributions

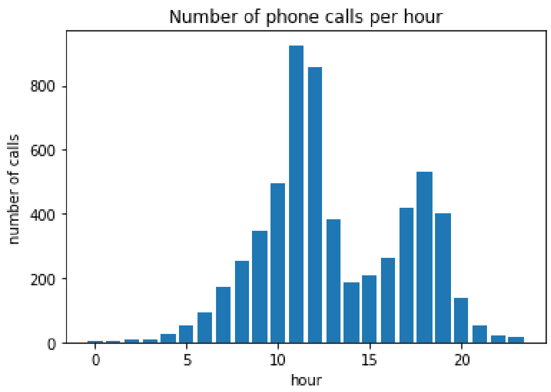

Beta Distributions

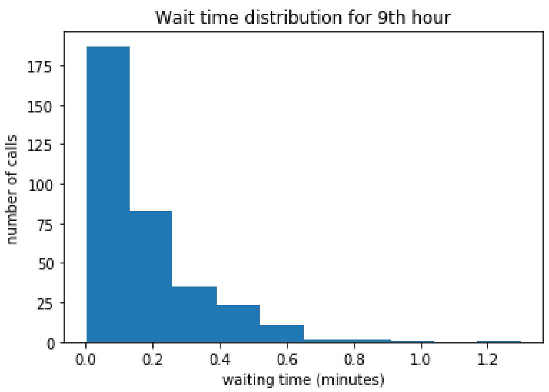

Gamma Distribution: k=2.0, θ=1.0

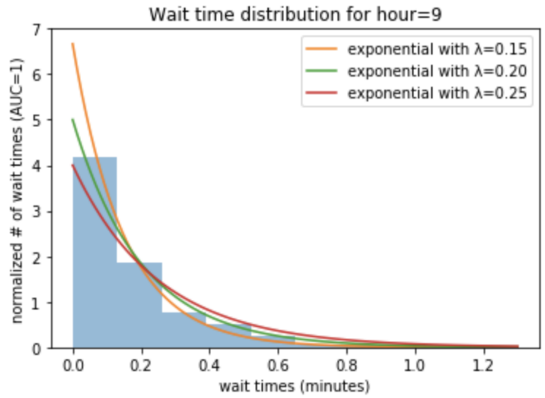

Normal Distribution: μ=0.0, σ=1.0


Beta Distribution: μ=6.0, σ=3.0

Back to Top







```
KS-Test results (goodness of fit):
lambda D-statistic p-value
 0.15  0.066       9.098e-02
  0.2  0.088       9.084e-03
 0.25  0.163       1.656e-08
```

Study guide

- **Scipy.stats** distributions
  - stats.**gamma**(a=α, scale=θ=1/β)
    - α: **shape** parameter: k-th event that one wants to observe
    - β: **rate** parameter
    - θ: **scale** parameter = 1/β = **mean wait time**
    - $p(x \mid \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$
    - Wait times between **k+1** Poisson distributed events
  - stats.**norm**(loc=μ, scale=σ)
    - μ: mean
    - σ: standard deviation
    - $p(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp \frac{-(x-\mu)^2}{2\sigma^2}$
  - stats.**beta**(a=α, b=β)
    - α, β: **shape** parameters
    - $p(x \mid \alpha, \beta) = \frac{1}{B(\alpha,\beta)} x^{\alpha-1} (1-x)^{\beta-1}$
  - stats.**expon**(scale=1/lambda_)

Back to Top

        - λ:
        - $p(x \mid \lambda) = \lambda e^{-\lambda x}$
        - Wait times between **two** Poisson distributed events
- **Scipy.stats** distributions functions
    - **Pdf**: probability density function
    - **Cdf**: cumulative distribution function
    - **Ppf:** percent point function (i.e. quantile function)
    - **Rvs**: random sample

Readings

Pyplot tutorial

https://matplotlib.org/tutorials/introductory/pyplot.html
This is a tutorial for using the plotting functionality in Matplotlib, a Python library. If you have never done plotting in Python before, this is a good tutorial to work through. There is more information about plotting in the study guide, which focuses on how to plot histograms and pdfs.

-

Statistical functions, scipy.stat

https://docs.scipy.org/doc/scipy/reference/stats.html
This reference contains all the common parametric distributions available in Python and how to evaluate and sample from them as well as the cumulative distribution and quantile functions. You are not expected to read through all of this, but use this reference together with the study guide to learn how to find the Python functions for the distributions you need.
If you find it difficult to trawl through all this documentation, there is also a scipy.stats cheat sheet available at the link below. This cheat sheet is not exhaustive but contains the common, useful distributions for the course. Let your instructor know if you would like to add more distributions.
https://docs.google.com/document/d/1FI-pfYTlZ9W7soS-4B6KxBAsueWlM-8XAlIIx09kLF4/edit

-

Lambert, B. (2018, May 16). *An introduction to the **gamma distribution***

We use the gamma distribution in class today and will use it again during the remainder of the course. Focus on how the alpha and beta parameters of the distribution affect its shape. When you need to choose a prior distribution for a model, it helps a lot to know how you can "shape" a distribution by controlling its parameters.
The derivation of the mean of the gamma distribution, from 12:00 onwards, is optional.

- $y \sim gamma(\alpha, \beta)$
    - $p(x \mid \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$
    - $\Gamma(n) = (n-1)!$ (i.e. simple factorial)
    - $\Gamma(z) = \int_0^\infty \quad x^{z-1} e^{-x} dx, R(z) > 0$(factorial to complex numbers except non-positive integers)
    - **Continuous** distribution

  - Defined only for **x >= 0** (i.e. **non-negative** x)
- Used for when:
  - Mean measure for a **count** variable (i.e. estimate for λ in a Poisson distribution)
  - Prior for a precision
- **α: shape parameter**
  - α=1: $p(x|\alpha = 1, \beta) \sim e^{-\beta x}$: exponential decay (**exponential** distribution is just gamma distribution with α=1)
  - α=2: $p(x|\alpha = 2, \beta) \sim xe^{-\beta x}$: 1st-order power * exponential decay
  - α=3: $p(x|\alpha = 3, \beta) \sim x^2e^{-\beta x}$: 2nd-order power * exponential decay
  - The larger the α, the further to the positive x-axis the PDF peaks
  - But exponential decay always wins over power increase
- **β: rate parameter**
  - $p(x|\alpha = C, \beta) \sim \beta^\alpha e^{-\beta x}$
  - Increase β: increase **peak height**, and **sharper decline**
    i.e. **taller & sharper**
- **Mean** of gamma distribution = α/β = $k\theta$
  - $$\begin{aligned} E[x] &= \int_0^\infty \quad x \cdot gamma(x \mid \alpha, \beta)\, dx \\ &= \int_0^\infty \quad x \cdot \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}\, dx \\ &= \frac{\beta^\alpha}{\Gamma(\alpha)} \int_0^\infty \quad x^\alpha e^{-\beta x}\, dx \\ &= \frac{\beta^\alpha}{\Gamma(\alpha)} \cdot \frac{\Gamma(\alpha+1)}{\beta^{\alpha+1}} \int_0^\infty \quad \frac{\beta^{\alpha+1}}{\Gamma(\alpha+1)} x^\alpha e^{\beta x}\, dx \\ &= \frac{\beta^\alpha}{\Gamma(\alpha)} \cdot \frac{\Gamma(\alpha+1)}{\beta^{\alpha+1}} \int_0^\infty \quad gamma(x \mid \alpha+1, \beta)\, dx \\ &= \frac{\beta^\alpha}{\Gamma(\alpha)} \cdot \frac{\Gamma(\alpha+1)}{\beta^{\alpha+1}} \because \text{all probability distributions integrate to 1} \\ &= \alpha/\beta \because \text{gamma function is a factorial for } (\alpha\text{-1}) \text{ when } \alpha \in \mathrm{N} \end{aligned}$$

Lambert, B. (2018, May 16). *An introduction to the beta distribution*

- $y \sim beta(a, b)$
  - $p(\theta \mid a, b) = \frac{\theta^{a-1}(1-\theta)^{b-1}}{B(a,b)}, \theta \in [0,1]$
  - But B(a,b) is a normalization constant that simply allows the probability distribution to integrate to 1
  - $\therefore\ p(\theta \mid a, b)\ \propto\ \theta^{a-1}(1-\theta)^{b-1}$
  - Defined only for $\boldsymbol{\theta}$ ∈ [0,1]
- Used for when:
  - Specifying **priors** for **probabilities**
  - **Conjugate prior** for Bernoulli, binomial, and geometric distributions
- Shape:
  - $a = b = 0.5$: $p(\theta|a, b) \propto \theta^{-1/2}(1-\theta)^{-1/2}$
    When both a and be are (0,1), p is asymptotic at both 0 and 1
  - $a = 1, b = 0.5$: $p(\theta|a, b) \propto \theta^{-1/2}$
    When one is (0,1) while the other is 1, p is a negative power function
  - $a = b = 1$: $p(\theta|a, b) \propto 1$
    When both are 1, p is constant

Back to Top

  - $a = b = 3 : p(\theta|a, b) \propto \theta^2(1-\theta)^2$
    When a,b>1, p is 0 at x=0,1 and has a peak
- Mean:
  - $E[\theta] = \frac{a}{a+b}$
  - When a > b, graph is slanted to the right and vice versa

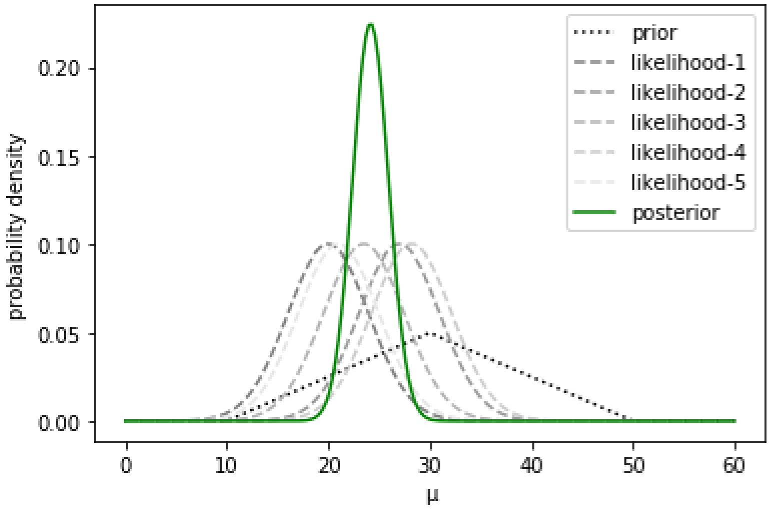
Class

Back to Top

CS146 2.1 - Bayesian Inference

Pre-class

Gist link: https://gist.github.com/AntonioStark/c6f0ccd608a25d3adf1a171dfec28d4c
PDF: https://drive.google.com/file/d/1C-M8nM7EV9rjxPIP6mLJUmkqe__cl-Qi/view?usp=sharing



Study guide

- θ: parameters
- p(θ): **prior distribution**: estimate/distribution of parameters before data
- p(θ | data): **posterior distribution**: estimate/distribution of parameters after data
- **Bayes' rule**: p(θ | data) = p(data | θ) * p(θ) / p(data)
- p(data|θ): **likelihood function**
    - Integrates to 1 over all possible values of **data** but not **parameters** (θ)
- "Creating a **statistical model**": selecting/designing an appropriate likelihood function

Readings

Lambert, B. (2018, May 15). *Why is a likelihood not a probability distribution?*

Focus on the differences between probability distributions and likelihood functions. You should be clear on which variables vary and which are constant in each case.

- When the **data** is held constant, but the **parameter** varies
    - We get a **likelihood distribution**
    - Whose integral **does not equal to 1**
    - **p(data|θ)**, or **L(θ|x=?)**, or **p(x=?|θ)=θ**
- Bayesian rule: converts **likelihood** (which does **not** describe uncertainty) into **probability** (which **does** describe uncertainty)

Back to Top

Lambert, B. (2018, May 15). *Why is it difficult to calculate the denominator of Bayes' rule in practice?*

It might not seem that important that it is difficult to compute the denominator in Bayes' equation but it is the main reason why we cannot do exact inference in most practical problems and have to use approximate methods instead. Understand why it is virtually impossible to compute the denominator in high-dimensional models.

- $p(\theta|X) = \frac{p(X|\theta)\,p(\theta)}{p(X)} = \frac{p(X,\theta)}{p(X)}$
- p(X|θ) is a valid distribution
- But **p(X)** (the denominator) is difficult to find
    - 1-dimensional discrete: $p(X) = \sum_{\theta=\theta_1}^{\theta_p} \quad p(X,\theta)$
    - 1-dimensional continuous: $p(X) = \int_{-\infty}^{\infty} \quad p(X,\theta)\, d\theta$
    - N-dimensional continuous: $p(X) = \int_{-\infty}^{\infty} \quad \int_{-\infty}^{\infty} \quad \cdots \int_{-\infty}^{\infty} \quad p(X,\theta,\phi,\dots,\epsilon)\, d\epsilon \cdots d\phi\, d\theta$
    - ⇒ we choose to **sample** rather than to (approximate deterministically) calculate posterior distributions

Lambert, B. (2018, May 15). Explaining the intuition behind Bayesian inference

Focus on how the posterior is a weighted average between the likelihood and the prior. The more data we have, the more important the likelihood becomes compared to the prior. Also, pay attention to the plots shown towards the end of the video. Notice how the posterior plot changes to interpolate the prior and the likelihood.

- $p(\theta|X) = \frac{p(X|\theta)\,p(\theta)}{p(X)} = \frac{p(X,\theta)}{p(X)}$

  ∝$p(X|\theta)\,p(\theta)$

  ⇒ **weighted average** of the **likelihood** & **prior** (i.e. p(θ|data) and p($\boldsymbol{\theta}$**))**
- **Posterior peak** is between those of **likelihood** & **prior**
- As the **amount of data** (X) increases,
    - Likelihood p(X|θ) becomes **narrower**
    - **Posterior peak** becomes closer to the **likelihood** than prior
    - Posterior peak also becomes **narrower**

Lambert, B. (2018, May 15). *An introduction to the Poisson distribution 1, 2*

Focus on the properties of the Poisson distribution and the scenarios for which the Poisson distribution is appropriate. The proof that the Poisson distribution sums to 1 is not important for today's session. More interesting properties of the Poisson distribution. We will discuss the "conjugate priors" of various distributions later in the course.

- **Poisson distribution**
    - **Discrete** distribution
    - $p(y|\lambda) = \lambda^y e^{-\lambda} \,/\, y!, y \in N$
    - Measure **counts** of
        - Events that occur **independently** at a **uniformly random rate** (λ)
    - The above is a **probability distribution** because sum=1

Back to Top

        - $\sum_y \quad \frac{\lambda^y e^{-\lambda}}{y!} = e^{-\lambda} \sum_y \quad \frac{\lambda^y}{y!} = e^{-\lambda} e^{\lambda} = 1$
          b/c Mclaurin expansion of $e^x = \sum_{k=0}^{\infty} \quad \frac{x^k}{k!}$
- Poisson distribution **mean = λ**
    - $E[y] = \sum_y \quad y * p(y|\lambda)$
- Poisson distribution **variance = λ**
- ⇒ use Poisson distribution when **mean ≈ variance**
    - When variance > mean ⇒ use negative binomial
- If
    - The prior, λ, is a **gamma distribution**
    - and **likelihood** is a **Poisson distribution**
    - Then the posterior will be a **gamma distribution**
    - I.e. λ=Gamma(α,β) is a **conjugate prior** when likelihood is Poisson of said λ
- **Binomial distribution**
    - $X \sim Bin(N, p)$
    - $p(X = k) = \frac{N}{k} p^k (1-p)^k$
    - $\frac{N}{k}$is difficult to calculate
    - Can be approximated when:
        - **N is large**
        - **p is small**
    - Approximation via
        - A Poisson distribution of λ=N*p

Rohrer, B. (2016, November 1). *How Bayes Theorem works*

This is a very gentle summary of how Bayes' theorem is used to update our prior knowledge or belief using measurements/data to get our posterior knowledge or belief. Watch the video if you want to revise some Bayesian inference basics.

- MLE: maximum likelihood estimate
- Maximum a posteriori

Class

Back to Top

CS146 1.2 - Probability Distributions

Pre-class

Gist: https://gist.github.com/AntonioStark/2dc0e624a6916b54c38241386c2628c6
Document: https://drive.google.com/file/d/1xy65Kfk6IFCUG43LExtuSaPCTE94Y6Y9/view?usp=sharing

Study guide

- **Support**: all the values that the random variable in the distribution can take
    - Normal distribution: R
    - Binomial distribution: N_0 (non-negative integers)
- **Mode**: value where the **probability density function** is maximum
- **CDF: cumulative distribution function**: distribution of the integral
- **QF: quantile function**: distribution of the inverse of the cdf

Readings

Chapter 23 of MacKay, D. J. C. (2003). Information theory, inference, and learning algorithms

Everybody has their favorite list of probability distributions. In this chapter, you will see some distributions you know and some unusual ones like the inverse-cosh and the Von Mises distributions. The goal is not to know all possible distributions (there are infinitely many of them), but to understand what a probability distribution is, how to explore its characteristics (like support, parameters, mode(s), mean, typical sample values). Plotting distributions is also very helpful. Look at study guide and pre-class work for exploring and characterizing some of the distributions in this chapter.

- 23.1 Distributions over **integers**
    - **Binomial**
        - Parameters
            - **Bias** (f or p)
            - **# of trials** (N)
        - For when:
            - Each trial has **two outcomes**
            - Each trial is **independent**
            - **Probability of success** in each trial is constant
        - Equations
            - Probability of getting a certain # of successes:

$$P(r \mid f, N) = \binom{N}{r} f^r (1-f)^{N-r} \qquad r \in \{0, 1, 2, \ldots, N\}$$

$$\boldsymbol{P(x) = \frac{n!}{(n-x)!\,x!} p^x q^{n-x}}$$

            - Mean = n*p

Back to Top

      - Variance = var(x) = n*p*q = n*f*(1-f)
  - **Poisson**
    - Parameters
      - λ: rate of which event (or **success**) occurs
    - For when:
      - **Number of events** to occur in a specific **time interval**
      - Each event (or **success**) is independent
      - **Probability of success** remains constant
    - Equations
      - Probability that there will be *r* events in **1 time interval unit** (equation is modified as $\mu = \lambda$*t $= \lambda$

        $$P(r\,|\,\lambda) = e^{-\lambda}\frac{\lambda^r}{r!} \qquad r \in \{0, 1, 2, \ldots\}$$
  - **Exponential**
    - Parameters
      - f: the possibility of failure
    - For when:
      - Model the **time** elapsed between events (or amount of **waiting time**)
    - Equations
      - Probability of success happening at time=r:

        $$P(r\,|\,f) = f^r(1-f)$$

        $$P(r\,|\,f) = (1-f)\,e^{-\lambda r}$$

        (λ = ln(1/f))
- 23.2 Distributions over **unbounded real numbers**
  - **Gaussian normal**
    - Parameters
      - μ: mean
      - σ: standard deviation
    - Equations
      - Precision parameter

        $\tau = 1/\sigma^2$
      - P(x):

        $$P(x\,|\,\mu,\sigma) = \frac{1}{Z}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \qquad x \in (-\infty, \infty),$$
      - $$Z = \sqrt{2\pi\sigma^2}.$$
  - **Student (Student's T)**
    - Normal distribution for estimating means of samples when sample size (degree of freedom) is small
  - **Cauchy**
    - **Student** distribution with degree of freedom = 1
  - **Biexponential**
  - **Inverse-cosh**
- 23.3 Distributions over **positive real numbers**
  - **Exponential**
    - (explained above) for calculating **wait time** for the **first event**

Back to Top

- **Gamma**
    - For calculating **wait time** for the **k-th event**
    - Parameters
        - k: the # of events for which one is waiting
        - λ: **rate** of event
    - Equations
        - Mean = sc
        - Variance = s^2c
- **Inverse-gamma**
    - Reciprocal of the gamma distribution
    - Parameters
        - α: the height
        - β: the spread
- **Log-normal (Galton)**
    - If the data's **logarithm** is normally distributed
    - Parameters:
        - σ: shape parameter
        - M: scale parameter
        - μ: location parameter
    - Useful when:
        - Skewed, large variance distributions

- 23.4 Distributions over **periodic variables**
    - **Von Mises**
    - **Wrapped Gaussian**
- 23.5 Distributions over **probabilities**
    - ⇒ i.e. the **range** is confined to between 0 and 1
    - ⇒ i.e. a probability distribution over probabilities
    - **Beta**
        - Usually used to model **priors** for probabilities (before you make any observations) and updating them as one makes more observations
        - Parameters: α, β
    - **Dirichlet**
        - The **beta** distribution is a special case of the Dirichlet where I (# of dimensions) = 2
        - Also used to model priors in Bayesian statistics
    - **Entropic**

NIST/SEMATECH. (2012, April). What is a Probability Distribution, *e-Handbook of Statistical Methods*.

This reading outlines the definition of a probability distribution — not any specific distribution, but the general requirements that all distributions satisfy. You should know that any function that has a non-negative probability for each value of its variable (or variables) and where the sum or integral over probabilities of all values for which the function is defined is 1, is a probability distribution by definition. See the pre-class work for designing your own distribution using this definition.

- Properties of **discrete probabilities**

Back to Top

    - $\forall x, \exists P[X=x] = \rho(x) = \rho_x$
    - $\forall x, \rho(x) >= 0$
    - $\Sigma \rho_j = 1$
- Properties of **continuous probabilities**
    - $\forall x \in R, \rho(a \leq x \leq b) >= 0$
    - Integral of the function = 1
- Probability **mass vs. density** functions
    - **Probability mass function:** function that represents a **discrete** probability distribution
        - ⇒ probability = the **output** of the function
    - **Probability density function:** a function that represents a **continuous** probability distribution
        - ⇒ probability = the **area under the curve** of the function

(Optional) Khan Academy. (n.d.). *Random variables*

These Khan Academy videos contain background knowledge for this course, including an introduction to discrete and continuous probability distributions. Please watch these if you have some knowledge gaps about probabilities.

Class

Back to Top

CS146 1.1 - Modeling Under Uncertainty

Pre-class

There is no pre-class work for this session. This is a good opportunity to make progress with your first assignment, which is due at the end of the week. The first assignment is based on the course prerequisites, so if you haven't looked at those yet, do so now!

Study guide

You should be able to

- explain the meaning or interpretation of a probability in the context of data modeling and inference;
- compare the frequentist and Bayesian interpretations of probabilities.

Readings

Chapter 1 of McElreath, R. (2015). Statistical rethinking: A Bayesian course with examples in R and Stan. Chapman and Hall/CRC Press

Depending on how you learned about probability theory and statistics before, you might think of statistical modeling as a **collection of different hypothesis and significance tests** that can be applied to data from repeatable experiments, or **unified theory of how data can be used** to update estimates of parameters of interest probabilistic models. This chapter gives some of the history and philosophical thought behind these two frameworks, which we will explore in a lot more detail during this course.

- 1.1 Statistical golems
  - **Statistics**: neither mathematics nor science, but **engineering**
  - Classical tools: not diverse enough to handle many common research questions
- 1.2 Statistical rethinking
  - **Popperism** (Karl Popper): science advances by **falsifying hypotheses** (i.e. **deductive falsification)**
    - **Hypotheses are not models**. Many models correspond to the same hypotheses while many hypotheses can correspond to a single model ⇒ strict falsification is impossible
    - **Measurement** matters. Our data may be wrong
  - **NHST: null-hypothesis significance testing**: different
  - 1.2.1 Hypotheses are not models
    - All hypothesis is from a model (models **operationalize** hypotheses)
    - **Statistical models** routinely correspond to multiple **detailed process models** b/c they rely on distributions
  - 1.2.2 Measurement matters
    - **Modus tollens** (method of destruction): concludes that H is false if we do not find D, but does not tell anything about H if we do find D

Back to Top

-
  -
    - 1.2.2.1 **Observation error**
    - 1.2.2.2 **Continuous hypotheses**
      - Many scientific hypotheses are **probablistic**, not of strict differences
  - 1.2.3 **Falsification** is **consensual** not **logical**
- 1.3 Three tools for golem engineering
  - 1.3.1 **Bayesian data analysis**
    - **Frequentist** approach: all probabilities defined by countable events & their frequencies among large samples
      - ⇒ parameters & models **cannot** have probability distributions
      - ⇒ ⇒ only **measurements** can have probability distributions
      - ⇒ ⇒ ⇒ **sampling distribution**
      - ⇐ cannot explain Saturn's rings by Galileo - his uncertainty is not from **sampling variation**
    - **Bayesian** approach: randomness only describes our **incomplete knowledge** - the reality is deterministic
  - 1.3.2 **Multilevel models**
    - A.k.a. Hierarchical, random effects, varying effects, mixed effects model
    - Four reasons to use multilevel models:
      - Adjust estimates for **repeat sampling**
      - Adjust estimates for **sampling imbalance**
      - Study **variation**: multilevel models model variation explicitly
      - Avoid **averaging**
    - Good when there are **clusters/groups**
    - **Multilevel regression** should be the **default regression**
  - 1.3.3 Model comparison and information criteria
    - **Information criteria**
      - E.g. **AIC: Akaike information criterion**
    - Solves these difficulties of model comparison:
      - **Overfitting**: "fitting is easy, prediction is hard"
        - Information criteria provide **predictive accuracy**
      - **Comparison of multiple non-null models to same data**
- 1.4 Summary

McElreath, R. (2017, April 22). Bayesian statistics without frequentist language

This video by the author of the chapter above provides more insight into how we use terminology in the course. You will see two views of Bayesian statistical modeling presented in this video – called the "inside" and the "outside" view – in the course and in books and papers on statistical modeling. It is important that you understanding the technical language since we will use it constantly during the course. The video also covers some Bayesian models and the implementation of those models in a language called Stan. You do not have to understand these parts in detail for today's class. Try to get an overview of what it means to build and solve statistical models. Much of this course is dedicated to this modeling process and you will learn about it in detail in the weeks to come.

- **Outside view**
  - Data have distributions

Back to Top

    - Parameters do not have distributions
    - Parmaters != statistics
    - Likelihood != probability distribution
    - **Imaginary** population
    - Bayes: sampling theory + priors
    - Priors: uniquely subjective
- Conceptual friction
    - Data must look like **likelihood functions**
    - **Degrees of freedom**
    - **Sampling** is the source of all uncertainty
    - Defining random effects via sampling design
    - Neglect **data uncertainty**
- **Inside view**
    - Bayesian approach: joint generative model of all variables
    - **Unity among variables:** no distinction between **data** vs. **parameters**
    - **Unity among distributions:** no distinction between **likelihood** vs. **priors**
- **Likelihood** vs. **prior**
    - B = Normal (u, s)
    - If b is observed, likelihood
    - If b is unobserved, prior
- How is a **prior** formed?
    - **Maximum entropy**: finds the distribution that embodies certain characteristics and nothing more (i.e. **maximally conservative distribution**)
- GLM: generalized linear model
- GLMM: generalized linear mixed models
- **Shrinkage** happens everywhere where a distribution that's a function of parameters
    - ⇒ **regression towards the mean**
    - ⇒ shrinkage of the parameters (not the data)
- The advantage of **inside** view over **outside** view of Bayesian models:
    - ⇒ it treats **data & parameters, likelihoods & priors** in the same way
    - ⇐ they have the same derivations & calculations (inside computers)
    - ⇐ they have same **inferential force**, i.e. **shrinkage** (i.e. greater skepticism for more extreme values)
    - ⇐ the same symbols can be both data/parameters in the same model and analysis
- Benefits of insider view
    - Not necessary but useful
    - Think scientifically, not statistically
        - Can account for missingness and uncertainty
- Proposed terminology changes
    - Data → observed variable
    - Parameter → unobserved variable
    - Likelihood → distribution
    - Prior → distribution
    - Posterior → conditional distribution

Back to Top

    - `Estimate, random → `*banished*

Class

- Bayesian vs. frequentist probability
    - **Frequentist**: data varies, parameters are fixed
    - **Bayesian**: data is fixed, the parameters varies
- Likelihood vs. probability
    - **Probability**: areas under fixed distribution
        - P(data | parameters)
    - **Likelihood**: y-axis points for fixed data points with varying distributions
        - L(parameters | data)
    - **Prior**: probability of parameters
- Prediction vs. inference
    - **Inference**
        - **Modeling**: Reason about the data generation process and choose the stochastic model that approximates the **data generation process** best.
        - Model **validation**: Evaluate the validity of the stochastic model using residual analysis or goodness-of-fit tests.
        - **Inference**: Use the stochastic model to understand the data generation process
    - **Prediction**
        - **Modeling**: Consider several different **models** and different **parameter** settings.
        - Model **selection**: Identify the model with the greatest predictive performance using validation/test sets; select the model with the highest performance on the test set.
        - **Prediction**: Apply the selected model on new data with the expectation that the selected model also generalizes to the unseen data.
-