Contents

Probabilistic Graphical Models

Contents

Probabilistic Graphical Models **May 13, 2018**

Foundations (Ch. 2)

Table of Contents Local *Written by Brandon McKinzie*

(Brief summary of the book's notation, for ease of reference.)

Graphs. Authors denote directed graphs as $\mathcal{G}$ and undirected graphs as $\mathcal{H}$.

- **Induced Subgraph**. Let $\mathcal{K} = (\mathcal{X}, \mathcal{E})$ and $\boldsymbol{X} \subset \mathcal{X}$. Define the *induced subgraph* $\mathcal{K}[\boldsymbol{X}]$ to be the graph $(\boldsymbol{X}, \mathcal{E}')$ where $\mathcal{E}'$ are all edges $X \rightleftharpoons Y$ such that $X, Y \in \boldsymbol{X}$.
- **Complete Subgraph**. A subgraph over $\boldsymbol{X}$ is *complete* if every two nodes in $\boldsymbol{X}$ are connected by some edge. The set $\boldsymbol{X}$ is often called a *clique*; we say that a clique $\boldsymbol{X}$ is *maximal* if for any superset of nodes $\boldsymbol{Y} \supset \boldsymbol{X}$, $\boldsymbol{Y}$ is not a clique.
- **Upward Closure**. We say that a subset of nodes $\boldsymbol{X} \in \mathcal{X}$ is *upwardly closed* in $\mathcal{K}$ if $\forall X \in \boldsymbol{X}$, we have that $\text{Boundary}_X \subset \boldsymbol{X}$[1]. We define the *upward closure* of $\boldsymbol{X}$ to be the minimally upwardly closed subset $\boldsymbol{Y}$ that contains $\boldsymbol{X}$. We define the *upwardly closed subgraph* of $\boldsymbol{X}$, denoted $\mathcal{K}^+[\boldsymbol{X}]$, to be the induced subgraph over $\boldsymbol{Y}$, $\mathcal{K}[\boldsymbol{Y}]$.

Paths and Trails. Definitions for longer-range connections in graphs. We use the notation $X_i \rightleftharpoons X_{j+1}$ to denote that X_i and X_j are connected via some edge, whether directed (in any direction) or undirected.

- **Trail/Path**. We say that $X_1, \ldots, X_k$ form a *trail* in the graph $\mathcal{K} = (\mathcal{X}, \mathcal{E})$ if $\forall i = 1, \ldots, k-1$, we have that $X_i \rightleftharpoons X_{j+1}$. A *path* makes an additional restriction: either $X_i \rightarrow X_{i+1}$ or $X_i — X_{i+1}$.
- **Connected Graph**. A graph is *connected* if $\forall X_i, X_j$ there is a trail between X_i and X_j.
- **Cycle**. A *cycle* in $\mathcal{K}$ is a directed path $X_1, \ldots X_k$ where $X_1 = X_k$.
- **Loop**. A *loop* in $\mathcal{K}$ is a trail where $X_1 = X_k$. A graph is *singly connected* if it contains no loops. A node in a singly connected graph is called a *leaf* if it has exactly one adjacent node.
- **Polytree/Forest**. A singly connected graph is also called a *polytree*. A singly connected undirected graph is called a *forest*; if a forest is also connected, it is called a *tree*.
  - A directed graph is a forest if each node has at most one parent. A directed forest is a tree if it is also connected.
- **Chordal Graph**. Let $X_1 — X_2 — \cdots — X_k — X_1$ be a loop in the graph. A *chord* in the loop is an edge connecting X_i and X_j for two nonconsecutive nodes X_i, X_j. An undirected graph $\mathcal{H}$ is said to be *chordal* if any loop $X_1 — X_2 — \cdots — X_k — X_1$ for $k \geq 4$ has a chord.

[1] $\text{Boundary}_X \triangleq \text{Pa}_X \cup \text{Nb}_X$. For DAGs, this is simply X's parents, and for undirected graphs X's neighbors.

Probability. Some notational reminders for this book. Let Ω denote a space of possible outcomes, and let S denote a set of measurable **events** α, each of which are a subset of Ω.

> *A probability distribution P over (Ω, S) is a mapping from events in S to real values that satisfy:*
>
> - $P(\alpha) \geq 0$ *for all* $\alpha \in S$.
> - $P(\Omega) = 1$.
> - *If* $\alpha, \beta \in S$ *and* $\alpha \cap \beta = \varnothing$, *then* $P(\alpha \cup \beta) = P(\alpha) + P(\beta)$.

Some useful independence properties:

$$\begin{aligned}
\textbf{Symmetry}: &\quad (X \perp Y \mid Z) \implies (Y \perp X \mid Z) && (1)\\
\textbf{Decomposition}: &\quad (X \perp (Y, W) \mid Z) \implies (X \perp Y \mid Z) && (2)\\
\textbf{Weak Union}: &\quad (X \perp (Y, W) \mid Z) \implies (X \perp Y \mid Z, W) && (3)\\
\textbf{Contraction}: &\quad (X \perp W \mid Z, Y) \& (X \perp Y \mid Z) \implies (X \perp Y, W \mid Z) && (4)
\end{aligned}$$

My Proofs: Independence Properties

I'll be using the definition that $(X \perp Y \mid Z) \Leftrightarrow P(X, Y \mid Z) = P(X \mid Z)P(Y \mid Z)$. Given this definition the proof for the symmetry property is trivial. In what follows, I'll assume the LHS of the given implication is true, and then show that the RHS must hold as well.

Decomposition:

$$P(X, Y \mid Z) = \sum_w P(X, Y, w \mid Z) = \sum_w P(X \mid Z)P(Y, w \mid Z) = P(X \mid Z)P(Y \mid Z) \quad \checkmark$$

Weak Union:

$$\begin{aligned}
P(X, Y \mid Z, W) &= \frac{P(X, Y, W \mid Z)}{P(W \mid Z)} && (5)\\
&= \frac{P(X \mid Z)P(Y, W \mid Z)}{P(W \mid Z)} && (6)\\
&= \frac{P(X \mid Z)P(W \mid Z)P(Y \mid Z, W)}{P(W \mid Z)} && (7)\\
&= P(X \mid Z, W)P(Y \mid Z, W) \quad \checkmark && (8)
\end{aligned}$$

Contraction:

$$\begin{aligned}
P(X, Y, W \mid Z) &= P(Y \mid Z)P(X, W \mid Z, Y) && (9)\\
&= P(Y \mid Z)P(X \mid Z, Y)P(W \mid Z, Y) && (10)\\
&= P(X \mid Z)\left[P(Y \mid Z)P(W \mid Z, Y)\right] && (11)\\
&= P(X \mid Z)P(Y, W \mid Z) \quad \checkmark && (12)
\end{aligned}$$

We now define what "positive distribution" means, and a useful property of such distributions.

> *A distribution P is said to be* ***positive*** *if for all events $\alpha \in S$, such that $\alpha \neq \varnothing$, we have that $P(\alpha) > 0$.*

For positive distributions, and for mutually disjoint sets $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}, \boldsymbol{W}$, the **intersection** property also holds:

$$\textbf{Intersection}: \qquad (\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z}, \boldsymbol{W}) \& (\boldsymbol{X} \perp \boldsymbol{W} \mid \boldsymbol{Z}, \boldsymbol{Y}) \implies (\boldsymbol{X} \perp \boldsymbol{Y}, \boldsymbol{W} \mid \boldsymbol{Z}) \tag{13}$$

1.1.1 Appendix

Figured this would be a good place to put some of the definitions in the Appendix, too.

Information Theory (A.1).

- **Entropy**. Let $P(X)$ be a distribution over a random variable X. The *entropy* of X is defined as.

We treat $0 \log \frac{1}{0} = 0$

$$H_P(X) = \mathbb{E}_P \left[\lg \frac{1}{P(X)} \right] = \sum_x P(X = x) \lg \frac{1}{P(X = x)} \tag{14}$$

$$0 \leq H_P(X) \leq \lg |Val(X)| \tag{15}$$

  $H_p(X)$ is a lower bound for the expected number of bits required to encode instances sampled from $P(X)$. Another interpretation is that the entropy is a measure of our uncertainty about the value of X.
- **Conditional Entropy**. The *conditional entropy* of X given Y is

$$H_P(X \mid Y) = H_P(X, Y) - H_P(Y) = \mathbb{E}_P \left[\lg \frac{1}{P(X \mid Y)} \right] \tag{16}$$

$$H_P(X \mid Y) \leq H_P(X) \tag{17}$$

  which captures the additional cost (in bits) of encoding X when we're already encoding Y.
- **Mutual Information**. The *mutual information* between X and Y is

$$I_P(X; Y) = H_P(X) - H_P(X \mid Y) = \mathbb{E}_P \left[\lg \frac{P(X \mid Y)}{P(X)} \right] \tag{18}$$

  which captures how many bits we save (on average) in the encoding of X if we know the value Y.
- **Distance Metric**. A *distance metric* is any distance measure d evaluating the distance between two distributions that satisfies all of the following properties:
  - **Positivity**: $d(P, Q) \geq 0$ and $d(P, Q) = 0$ if and only if $P = Q$.

- **Symmetry**: $d(P, Q) = d(Q, P)$.
- **Triangle inequality**: For any three distributions P, Q, R, we have that

$$d(P, R) \leq d(P, Q) + d(Q, R) \tag{19}$$

- **Kullback-Liebler Divergence**. Let P and Q be two distributions over random variables $X_1, \ldots X_n$. The relative entropy, or KL-divergence, of P and Q is

$$D(P \| Q) = \mathbb{E}_P \left[\lg \frac{P(X_1, \ldots, X_n)}{Q(X_1, \ldots, X_n)} \right] \tag{20}$$

  Note that this only satisfies the positivity property, and is thus not a true distance metric.

Algorithms and Algorithmic Complexity (A.3).

- **Decision Problems**. A decision problem Π is a task that accepts an input (instance) ω and decides whether it satisfies a certain condition or not. The **SAT** problem accepts a formula in propositional logic and decides whether there is an assignment to the variables in the formula such that it evaluates to true. **3-SAT** restricts this to accepting only formulas in conjunctive normal form (CNF), and further restricted s.t. each clause contains at most 3 literals.
- **P**. A decision problem is in the class $\mathcal{P}$ if there exists a deterministic algorithm that takes an instance ω and determines whether $\omega \in \mathcal{L}_\Pi$ (the set of instances for which a correct algorithm must return true), in polynomial time in the size of the input ω.
- **NP**. A *non-deterministic algorithm* takes the general form: (1) nondeterministically guess some assignment γ to the variables of ω, (2) deterministically verify whether γ satisfies the condition of the problem.The algorithm will repeat these steps until it produces a γ that satisfies the problem. A decision problem Π is in the class $\mathcal{NP}$ if there exists a nondeterministic algorithm that accepts ω if and only if $\omega \in \mathcal{L}_\Pi$, and if the verification stage can be executed in polynomial time in the length of ω.
- **NP-hard**. Π is $\mathcal{NP}$-hard if for every DP $\Pi' \in \mathcal{NP}$, there is a polynomial-time transformation of inputs such that an input for Π' belongs to $\mathcal{L}_{\Pi'}$ if and only if the transformed instance belongs to $\mathcal{L}_\Pi$. Note that $\mathcal{NP}$-hard is a superset of $\mathcal{NP}$.

The SAT problem is $\mathcal{NP}$-hard.

- **NP-complete**. A problem Π is said to be $\mathcal{NP}$-complete if it is both $\mathcal{NP}$-hard and in $\mathcal{NP}$.

Combinatorial Optimization and Search (A.4). Below, I'll outline some common search algorithms. These are designed to address the following task:

> *Given initial candidate solution σ_{cur}, a score function* `score`*, and a set of* ***search operators*** *$\mathcal{O}$, search for the optimal solution σ_{best} that maximizes the value of $score(\sigma_{best})$*

Greedy local search (Algorithm A.5)

Repeat the following until $didUpdate$ evaluates to $false$ at the end of an iteration.

1. Initialize $\sigma_{best} := \sigma_{cur}$.
2. Set $didUpdate := false$.
3. For each operator $o \in \mathcal{O}$, do:
    (a) Let $\sigma_o := o(\sigma_{best})$.
    (b) If σ_o is legal solution, and $score(\sigma_o) > score(\sigma_{best})$, reassign $\sigma_{best} := \sigma_o$, and set $didUpdate := true$.
4. If $didUpdate == true$, go back to step 2. Otherwise terminate and return σ_{best}.

Beam search (Algorithm A.7)

We are given a **beam width** K. Initialize our *beam*, the set of at most K solutions we are currently tracking, to $\{\sigma_{cur}\}$. Repeat the following until termination[a]:

1. Initial the set of successor states $H := \varnothing$.
2. For each solution $\sigma \in Beam$, and each operator $o \in \mathcal{O}$, insert a candidate successor state $o(\sigma)$ into H.
3. Set $Beam := KBestScore(H)$[b].

Once termination is reached, return the best solution σ_{best} in $Beam$.

[a]Termination condition could be e.g. an upper bound on number of iterations or on the improvement achieved in the last iteration.

[b]Notice that this implies an underlying assumption of beam search: all successor states $\sigma \in H$ have scores greater than any of the states in the current beam. We always assume improvement.

Continuous Optimization (A.5).

- **Line Search**. Method for adaptively choosing the step size (learning rate) η at each training step. Assuming we are doing gradient ascent. We'd usually set the parameters θ at step $t+1$ to $\theta^{(t)} + \eta \nabla f(\theta^{(t)})$. Line search modifies this by instead defining the "line" $g(\eta)$ below, and searching for the optimal value of η along that line.

$$g(\eta) = \overrightarrow{\theta}^{(t)} + \eta \nabla f(\theta^{(t)}) \tag{21}$$

$$\eta^{(t)} = \arg\max_{\eta} [g(\eta)] \tag{22}$$

$$\theta^{(t+1)} \leftarrow \theta^{(t)} + \eta^{(t)} \nabla f(\theta^{(t)}) \tag{23}$$

At risk of stating the obvious, $g(\eta)$ is referred to as a "line" because it's a function of the form `mx + b`.

Some notes from the textbook "Numerical Optimization" (chapters 8 and 9).

The BFGS Method (8.1). Begin the derivation by forming the following quadratic model of the objective function f at the current iterate[2] θ_t:

For $m_t(p)$, p denotes the deviation at step t from the current parameters θ_t.

$$m_t(p) = f_t + \nabla f_t^T p + \frac{1}{2} p^T B_t p \tag{24}$$

where B_t is an $n \times n$ symmetric p.d. matrix that will be revised/updated every iteration (it is *not* the Hessian!). The minimizer p_t of this function can be written explicitly

$$p_t = -B_t^{-1} \nabla f_t \tag{25}$$

$\frac{\partial}{\partial p} \frac{1}{2} p^T B_t p = B_t p$

is used as the search direction, and the new iterate is

$$\theta_{t+1} \leftarrow \theta_t + \alpha_t p_t \tag{26}$$

where the step length α_t is chosen to satisfy the **Wolfe conditions**[3]. This is basically Newton's method with line search, except that we're using the *approximate Hessian* B_t instead of the true Hessian. It would be nice if we could somehow avoid recomputing B_t at each step. One proposed method involves imposing conditions on B_{t+1} based on the previous step(s). Require that ∇m_{t+1} equal ∇f at the latest two iterates θ_t and θ_{t+1}. Formally, the two conditions can be written as

$$\nabla m_{t+1}(-\alpha_t p_t) = \nabla f_{t+1} - \alpha_t B_{t+1} p_t = \nabla f_t \tag{29}$$
$$\nabla m_{t+1}(0) = \nabla f_{t+1} \tag{30}$$

We can rearrange the first condition to obtain the **secant equation**:

$$H_{t+1} y_t = s_t \quad \text{where} \tag{31}$$
$$H_{t+1} \triangleq B_{t+1}^{-1} \tag{32}$$
$$s_t \triangleq \theta_{t+1} - \theta_t \tag{33}$$
$$y_t \triangleq \nabla f_{t+1} - \nabla f_t \tag{34}$$

which is true only if s_t and y_t satisfy the **curvature condition**, $s_t^T y_t > 0$. The curvature condition is guaranteed to hold if we impose the Wolfe conditions on the line search. As is, this

[2]Recall that an "iterate" is just some variable that gets iteratively computed/updated. Fancy people with fancy words.

[3]The Wolfe conditions are the following sufficient decrease and curvature conditions for line search:

$$f(\theta_t + \alpha_t p_t) \le f(\theta_t) + c_1 \alpha_t \nabla f_t^T p_t \tag{27}$$
$$\nabla f(\theta_t + \alpha_t p_t)^T p_t \ge c_2 \nabla f_t^T p_t \tag{28}$$

for some constant $c_1 \in (0, 1)$ and $c_2 \in (c_1, 1)$.

still has infinitely many solutions for H_{t+1}. To determine it uniquely, we impose the additional condition that H_{t+1} is the closest of all possible solutions to the current H_t:

$$\min_H ||H - H_t||_W \quad \text{s.t.} \quad H = H^T, \; Hy_t = s_t \tag{35}$$

where $||\cdot||_W$ is the *weighted Frobenius norm*[4], and W can be any matrix satisfying $Ws_t = y_t$. For concreteness, assume that $W = \widetilde{G}_t$, where

$$\widetilde{G}_t = \int_0^1 \nabla^2 f(\theta_t + \tau\alpha_t p_t) \mathrm{d}\tau \tag{38}$$

The unique solution to H_{t+1} is given by

$$\textbf{(BFGS)} \qquad H_{t+1} = (I - \rho_t s_t y_t^T) H_t (I - \rho_t y_t s_t^T) + \rho_t s_t s_t^T \tag{8.16}$$

where $\rho_t = 1/(y_t^T s_t)$. The BFGS is summarized in algorithm 8.1 below.

Algorithm 8.1 (BFGS Method). Given starting point θ_0, convergence tolerance $\epsilon > 0$, and inverse Hessian approximation H_0. Initialize $t = 0$. While $||\nabla f_t|| > \epsilon$ do:

1. Compute search direction $p_t = -H_t \nabla f_t$.
2. Set $\theta_{t+1} = \theta_t + \alpha_t p_t$, where α_t is computed via line search to satisfy the Wolfe conditions.
3. Define $s_t = \theta_{t+1} - \theta_t$ and $y_t = \nabla f_{t+1} - \nabla f_t$.
4. Compute H_t by means of equation 8.16.
5. Increment t += 1 and go back to step 1.

[4]

$$||H||_W \triangleq ||W^{1/2} H W^{1/2}||_F \tag{36}$$

$$||C||_F^2 \triangleq \sum_{i,j} c_{ij}^2 \tag{37}$$

L-BFGS. Modifies BFGS to store a modified version of H_t implicitly, by storing some number m of vector pairs $\{s_i, y_i\}$, corresponding to the m most recent time steps. We use a recursive procedure to compute $H_t \nabla f_t$ given the set of vectors.

Algorithm 9.1 (L-BFGS two-loop recursion) Subroutine of L-BFGS for computing $H_t \nabla f_t$. We're given the current value of ∇f_t, and we initialize q to this value.

1. For i in the range $[t-1, t-m]$, compute

$$\alpha_i \leftarrow \rho_i s_i^T q \tag{39}$$

$$q \leftarrow q - \alpha_i y_i \tag{40}$$

2. Set $r \rightarrow H_t^0 q$.
3. For i in the range $[t-m, t-1]$, compute

$$\beta \leftarrow \rho_i y_i^T r \tag{41}$$

$$r \leftarrow r + s_i(\alpha_i - \beta) \tag{42}$$

4. Return result $H_t \nabla f_t = r$.

Algorithm 9.2 (L-BFGS). Given starting point θ_0, integer $m > 0$, and initial $t = 0$. Repeat the following until convergence.

1. Choose H_t^0. A popular choice is $H_t^0 := \gamma_t I$, where

$$\gamma_t \triangleq \frac{s_{t-1}^T y_{t-1}}{y_{t-1}^T y_{t-1}}$$

2. Compute $p_t \leftarrow -H_t \nabla f_t$ from Algorithm 9.1.
3. Compute $\theta_{t+1} \leftarrow \theta_t + \alpha_t p_t$, where α_t is chosen to satisfy the Wolfe conditions.
4. if $t > m$, discard $\{s_{t-m}, y_{t-m}\}$. Compute and save s_t and y_t.
5. Increment t += 1 and go back to step 1.

1.1.3 Exercises

Going through all the problems with a star for review.

Exercise 2.4

Let $\alpha \in S$ be an event s.t. $P(\alpha) > 0$. Show that $P(\cdot \mid \alpha)$ satisfies the properties of a valid probability distribution.

- Show $P(\beta \mid \alpha) \geq 0$ for all $\beta \in S$. By definition,

$$P(\beta \mid \alpha) = \frac{1}{P(\alpha)} P(\alpha \cap \beta) \tag{43}$$

  and since the full joint $P \geq 0$ and since $P(\alpha) > 0$, we have the desired result.
- Show $P(\Omega_\alpha) = 1$. Again, using just the definitions,

$$\begin{align}
P(\Omega_\alpha) &= \sum_{\beta \in S} P(\beta \mid \alpha) \tag{44}\\
&= \frac{1}{P(\alpha)} \sum_{\beta \in S} P(\alpha \cap \beta) \tag{45}\\
&= \frac{1}{P(\alpha)} \left(\sum_{\beta \in \alpha} P(\beta) + \sum_{\gamma \notin \alpha} P(\varnothing) \right) \tag{46}\\
&= \frac{1}{P(\alpha)} \left(P(\alpha) + 0 \right) \tag{47}\\
&= 1 \tag{48}
\end{align}$$

- Show, for any $\beta, \gamma \in S$, where $\beta \cap \gamma = \varnothing$, that $P(\beta \cup \gamma \mid \alpha) = P(\beta \mid \alpha) + P(\gamma \mid \alpha)$.

$$\begin{align}
P(\beta \cup \gamma \mid \alpha) &= \frac{1}{P(\alpha)} P((\beta \cup \gamma) \cap \alpha) \tag{49}\\
&= \frac{1}{P(\alpha)} P((\beta \cap \alpha) \cup (\gamma \cap \alpha)) \tag{50}\\
&= \frac{1}{P(\alpha)} \left(P(\beta \cap \alpha) + P(\gamma \cap \alpha) \right) \tag{51}\\
&= P(\beta \mid \alpha) + P(\gamma \mid \alpha) \tag{52}
\end{align}$$

Exercise 2.16: Jensen's Inequality

Let f be a concave function and P a distribution over a random variable X. Then

$$\mathbb{E}_P[f(X)] \leq f(\mathbb{E}_P[X]) \tag{53}$$

Use this inequality to prove the following 3 properties.

- $H_P(X) \leq \log |Val(X)|$. Let $f(u) := \lg(u)$ be our concave function.

$$\begin{aligned} H_P(X) \triangleq \mathbb{E}_P\left[\lg \frac{1}{P(X)}\right] &= \mathbb{E}_P[f(u)] && (54) \\ &\leq f(\mathbb{E}_p[u]) = f\left(\sum_x u(x)P(x)\right) = f(|Val(X)|) = \lg |Val(X)| && (55) \\ & && (56) \end{aligned}$$

- $H_P(X) \geq 0$.

$$\begin{aligned} -H_P(X) &= -\mathbb{E}_P\left[\lg \frac{1}{P(X)}\right] && (57) \\ &= \mathbb{E}_P[\lg P(X)] \leq 0 && (58) \end{aligned}$$

  since $0 \leq P(X = x) \leq 1 \; \forall x$ (any term in the expectation where $P(X = x) = 0$ is equal to 0, by definition).
- $D(P||Q) \geq 0$. Use the same idea as in the first proof, but let $u(x) = Q(x)/P(x)$.

$$D(P||Q) \triangleq \mathbb{E}_P[\lg(P(X)/Q(X))] = -\mathbb{E}_p[f(u)] \tag{59}$$

$$-\mathbb{E}_p[f(u)] \geq f\left(\sum_x u(x)P(x)\right) = f(1) = 0 \tag{60}$$

Probabilistic Graphical Models **May 06, 2018**

The Bayesian Network Representation (Ch. 3)

Table of Contents Local *Written by Brandon McKinzie*

Koller and Friedman (2009). The Bayesian Network Representation.
Probabilistic Graphical Models: Principles and Techniques.

Goal: *represent a joint distribution P over some set of variables $\mathcal{X} = \{X_1, \ldots, X_n\}$.* Consider the case where each X_i is binary-valued. A *single* joint distribution requires access to the probability for each of the 2^n possible assignments for $\mathcal{X}$. The set of all such possible joint distributions,

Understanding the exponential blowup.

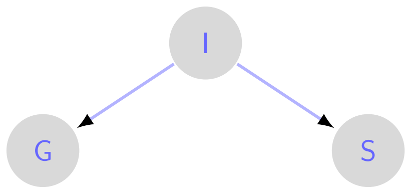
$$\{(p_1, \ldots, p_{2^n}) \in \mathbb{R}^{2^n} : \sum_{i=1}^{2^n} p_i = 1\} \tag{61}$$

is a $2^n - 1$ dimensional subspace of $\mathbb{R}^{2^n}$. Note that each p_i represents the probability for a unique instantiation of $\mathcal{X}$. Furthermore, in the general case, knowing p_i tells you nearly nothing about $p_{j \neq i}$ – i.e. you require an instantiation of $2^n - 1$ independent parameters to specify a given joint distribution.

Taking advantage of independencies.

But it would be foolish to parameterize any joint distribution in this way, since we can often take advantage of independencies. Consider the case where each X_i gives the outcome (H or T) of coin i being tossed. Then our distribution satisfies $(\boldsymbol{X} \perp \boldsymbol{Y})$ for any disjoint subsets fo the variables $\boldsymbol{X}$ and $\boldsymbol{Y}$. Let θ_i denote the probability that coin i lands heads. The key observation is that you only need each of the n θ_i to specify a unique joint distribution over $\mathcal{X}$, reducing the $2^n - 1$ dimensional subspace to an n dimensional manifold in $\mathbb{R}^{2^n}$[5].

The Naive Bayes Model. Say we want to determine the intelligence of an applicant based on their grade G in some course, and their score S on the SAT. A naive bayes model can be illustrated as below.



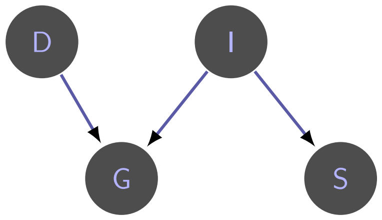
This encodes our general assumption that[6] $P \models (S \perp G \mid I)$.

[5]**TODO**: Formally define this manifold using the same notation as in 61. Edit: Not sure how to actually write it out, but intuitively it's because each of the 2^n p_i values, when going from the general case to the case of i.i.d, become *functions* of the n θ_i values. Whereas before they were independent free parameters.

[6]We say that an event α is independent of event β in P with the notation $P \models (\alpha \perp \beta)$. (Def 2.2, pg 23 of book)

In general, a naive bayes model assumes that instances fall into one of a number of mutually exclusive and exhaustive *classes*, defined as the set of values that the top variable in the graph can take on[7]. The model also includes some number of *features* $X_1, \dots X_k$, whose values are typically observed. The **naive Bayes assumption** is that the features are conditionally independent given the instance's class.

Bayesian Networks. A Bayesian network $\mathcal{B}$ is defined by a network structure together with its set of CPDs. **Causal reasoning** (or prediction) refers to computing the downstream effects of various factors (such as intelligence). **Evidential reasoning** (or explanation) is the reverse case, where we reason from effects to causes. Finally, **intercausal reasoning** (or explaining away) is when different causes of the same effect can interact. For our student example, we could be trying to determine $\Pr[I \mid G]$, the intelligence of an a student given his/her grade in a class. In addition to intelligence being a cause for the grade, we could have another causal variable D for the difficulty of the class:



An example of intercausal reasoning would be observing D, so that we now we want $\Pr[I \mid G, d]$; the diffulty of the course can help *explain away* good/bad grades, thus changing our value for the probability of intelligence based on the grade alone.

A **Bayesian network structure** $\mathcal{G}$ is a DAG whose nodes represent RVs $X_1, \dots X_n$. Let $Pa^{\mathcal{G}}_{X_i}$ denote the parents of X_i in $\mathcal{G}$, and NonDesc_{X_i} denotes the variables that are NOT descendants of X_i. Then $\mathcal{G}$ encodes the following set of ***local independencies***,

$$\mathcal{I}_\ell(\mathcal{G}) \triangleq \{\forall X_i : (X_i \perp \text{NonDesc}_{X_i} \mid Pa^{\mathcal{G}}_{X_i})\} \tag{62}$$

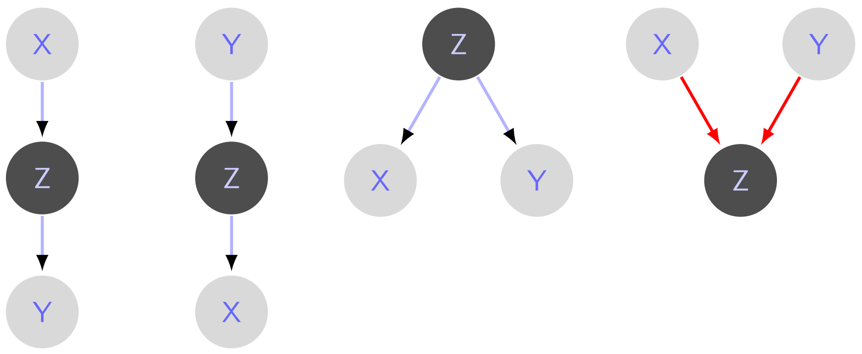
[7]For the intelligence example, the classes are high intelligence and low intelligence.

Graphs and Distributions. Here we see that a distribution P satisfies the local independencies associated with a graph $\mathcal{G}$ iff P is representable as a set of CPDs associated with $\mathcal{G}$.

- **Local independencies**. Let P be a distribution over $\mathcal{X}$. We define $\mathcal{I}(P)$ to be the set of **independence assertions** of the form $(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$ that hold in P. The statement "P satisfies the local independencies associated with $\mathcal{G}$" can thus be succinctly written:
$$\mathcal{I}_\ell(\mathcal{G}) \subseteq \mathcal{I}(P) \tag{63}$$
and we'd say that $\mathcal{G}$ is an **I-map** (independency map) for P.
- **I-maps**. More generally, let $\mathcal{K}$ be *any* graph object associated with a set of independencies $\mathcal{I}(\mathcal{K})$. We say that $\mathcal{K}$ is an **I-map** for a set of independencies $\mathcal{I}$ if $\mathcal{I}(\mathcal{K}) \subseteq \mathcal{I}$. Note that the complete graph (every two nodes connected) is an I-map for any distribution.
- **Factorization**. Let $\mathcal{G}$ be a BN graph over $X_1, \ldots, X_n$. We say that a distribution P over the same space **factorizes** according to $\mathcal{G}$ if P can be expressed as
$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid Pa^{\mathcal{G}}_{X_i}) \tag{64}$$
For BNs, the following equivalence holds:
$$\mathcal{G} \text{ is an I-map for } P \iff P \text{ factorizes according to } \mathcal{G}$$

D-separation. We want to understand when we can *guarantee* that an independence $(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$ holds in a distribution associated with a BN structure $\mathcal{G}$. Consider a 3-node network consisting of X, Y, and Z, where X and Y are not directly connected. There are four such cases, which I've drawn below.

The fourth trail is called a *v-structure*.



From left-to-right: indirect causal effect, indirect evidential effect, common cause, common effect. The first 3 satisfy $(X \perp Y \mid Z)$, but the 4th does not. Another way of saying this is that the first 3 trails are **active**[8] IFF Z is not observed, while the 4th trail is active IFF Z (or a descendent of Z) is observed.

[8]When influence can flow from X to Y via Z, we say that the trail $X \rightleftharpoons Y \rightleftharpoons Z$ is *active*.

General case:

> *Let $\mathcal{G}$ be a BN structure, and $X \rightleftharpoons \cdots \rightleftharpoons X_n$ a trail in $\mathcal{G}$. Let $\boldsymbol{Z}$ be a subset of observed variables. The trail is* ***active*** *given $\boldsymbol{Z}$ if*
>
> - *Any v-structure $X_{i-1} \to X_i \leftarrow X_{i+1}$ has X_i or one of its descendants in $\boldsymbol{Z}$.*
> - *No other node along the trail is in $\boldsymbol{Z}$.*

D-separation: (Directed separation)

> *Let $\boldsymbol{X}$, $\boldsymbol{Y}$, $\boldsymbol{Z}$ be three sets of nodes in $\mathcal{G}$. We say that $\boldsymbol{X}$ and $\boldsymbol{Y}$ are* ***d-separated*** *in $\mathcal{G}$, denoted d-sep$_{\mathcal{G}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z})$, if there is no active trail between any node $X \in \boldsymbol{X}$ and $Y \in \boldsymbol{Y}$ given $\boldsymbol{Z}$. We use $\mathcal{I}(\mathcal{G})$ to denote the set of independencies that correspond to d-separation,*
>
> $$\mathcal{I}(\mathcal{G}) = \{(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z}) : \textit{d-sep}_{\mathcal{G}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z})\} \tag{65}$$
>
> *also called the set of* ***global Markov independencies****.*

Soundness and Completeness of d-separation as a method for determining independence.

- **Soundness** (Thm 3.3). If a distribution P factorizes according to $\mathcal{G}$, then $\mathcal{I}(\mathcal{G}) \subseteq \mathcal{I}(P)$.
- **Completeness**. For any distribution P that factorizes over $\mathcal{G}$, we have that P is **faithful**[9] to $\mathcal{G}$:

$$X \perp Y \mid \boldsymbol{Z} \in \mathcal{I}(P) \implies \text{d-sep}_{\mathcal{G}}(X; Y \mid \boldsymbol{Z}) \tag{66}$$

To see the detailed algorithm for finding nodes reachable from X given $\boldsymbol{Z}$ via active trails, see Algorithm 3.1 on pg. 75 of the book.

[9] P is faithful to $\mathcal{G}$ if, whenever $X \perp Y \mid \boldsymbol{Z} \in \mathcal{I}(P)$, then d-sep$_{\mathcal{G}}(X; Y \mid \boldsymbol{Z})$.

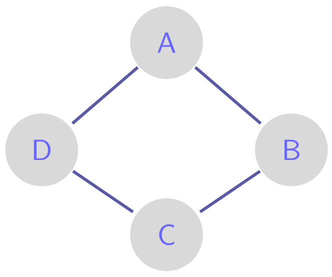
Probabilistic Graphical Models **November 03, 2017**

Undirected Graphical Models (Ch. 4)

Table of Contents Local *Written by Brandon McKinzie*

Koller and Friedman (2009). Undirected Graphical Models.
Probabilistic Graphical Models: Principles and Techniques.

The Misconception Example. Consider a scenario where we have four students who get together in pairs to work on homework for a class. Only the following pairs meet: (Alice, Bob), (Bob, Charles), (Charles, Debbie), (Debbie, Alice). The professor misspoke in class, giving rise to a possible misconception among the students. We have four binary random variables, $\{A, B, C, D\}$, representing whether the student has the misconception (1) or not (0) [10]. Intuitively, we want to model a distribution that satisfies $(A \perp C \mid \{B, D\})$, and $(B \perp D \mid \{A, C\})$, but no other independencies[11]. Note that the interactions between variables seem symmetric here – students influence each other (out of the ones they have a pair with).



The nodes in the graph of a **Markov network** represent the variables, and the edges correspond to a notion of direct probabilistic interaction between the neighboring variables – an interaction that is not mediated by any other variable in the network. So, how should we parameterize our network? We want to capture the **affinities** between the related variables (e.g. Alice and Bob are more likely to agree than disagree).

> *Let $\boldsymbol{D}$ be a set of random variables. We define a **factor** ϕ to be a function from $Val(\boldsymbol{D})$ to $\mathbb{R}$. A factor is nonnegative if all its entries are nonnegative. $\boldsymbol{D}$ is called the **scope** of the factor, denoted $Scope[\phi]$.*

We restrict our attention to nonnegative factors.

The factors need not be normalized. Therefore, to interpret probabilities over factors, we must

[10]A student might not have the misconception if e.g. they went home and figured out the problem via reading the textbook instead.
[11]These independences cannot be naturally captured in a Bayesian (i.e. directed) network.

normalize it with what we'll call the **partition function**, Z:

$$\Pr[a,b,c,d] = \frac{1}{Z}\phi_1(a,b)\cdot\phi_2(b,c)\cdot\phi_3(c,d)\cdot\phi_4(d,a) \tag{67}$$

$$Z = \sum_{a,b,c,d}\phi_1(a,b)\cdot\phi_2(b,c)\cdot\phi_3(c,d)\cdot\phi_4(d,a) \tag{68}$$

Parameterization. Associating the graph structure with a set of parameters. We parameterize undirected graphs by associating a set of factors with it. First, we introduce the definition of **factor product**:

> *Let $\boldsymbol{X}$, $\boldsymbol{Y}$, and $\boldsymbol{Z}$ be three disjoint sets of variables, and let $\phi_1(\boldsymbol{X},\boldsymbol{Y})$ and $\phi_2(\boldsymbol{Y},\boldsymbol{Z})$ be two factors. We define the factor product $\phi_1\times\phi_2$ to be a factor $\psi : Val(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z}) \mapsto \mathbb{R}$ as follows:*
>
> $$\psi(\boldsymbol{X},\boldsymbol{Y},\boldsymbol{Z}) = \phi_1(\boldsymbol{X},\boldsymbol{Y})\cdot\phi_2(\boldsymbol{Y},\boldsymbol{Z}) \tag{69}$$

We use this to define an undirected parameterization of a distribution:

> *A distribution $P_{\boldsymbol{\Phi}}$ is a **Gibbs distribution** parameterized by a set of factors $\boldsymbol{\Phi} = \{\phi_1(\boldsymbol{D}_1,\ldots,\phi_K(\boldsymbol{D}_K))\}$ if it is defined as follows:*
>
> $$P_{\boldsymbol{\Phi}} = \frac{1}{Z}\widetilde{P}_{\boldsymbol{\Phi}}(X_1,\ldots,X_n) \tag{70}$$
>
> $$\widetilde{P}_{\boldsymbol{\Phi}}(X_1,\ldots,X_n) = \phi_1(\boldsymbol{D}_1)\times\phi_2(\boldsymbol{D}_2)\times\cdots\times\phi_K(\boldsymbol{D}_K) \tag{71}$$

where the authors (Koller and Friedman) have made a point to emphasize: **A factor is only one contribution to the overall joint distribution. The distribution as a whole has to take into consideration the contributions from all of the factors involved.** Now, we relate the parameterization of a Gibbs distribution to a graph structure.

> *A distribution $P_{\boldsymbol{\Phi}}$ with $\boldsymbol{\Phi} = \{\phi_1(\boldsymbol{D}_1,\ldots,\phi_K(\boldsymbol{D}_K))\}$ **factorizes** over a Markov network $\mathcal{H}$ if each $\boldsymbol{D}_k (k=1,\ldots,K)$ is a complete subgraph[12] of $\mathcal{H}$.*

The factors that parameterize a Markov network are often called **clique potentials**. Although it can be used without loss of generality, the parameterization using maximal clique potentials generally obscures structure that is present in the original set of factors. Below are some useful definitions that we will use often.

[12] A subgraph is complete if every two nodes in the subgraph are connected by some edge. The set of nodes in such a subgraph is often called a **clique**. A clique $\boldsymbol{X}$ is maximal if for any superset of nodes $\boldsymbol{Y}\supset\boldsymbol{X}$, $\boldsymbol{Y}$ is *not* a clique.

Factor reduction:

*Let $\phi(\boldsymbol{Y})$ be a factor, and $\boldsymbol{U} = \boldsymbol{u}$ an assignment for $\boldsymbol{U} \subseteq \boldsymbol{Y}$. Define the **reduction** of the factor ϕ to the context $\boldsymbol{U} = \boldsymbol{u}$, denoted $\phi[\boldsymbol{u}]$, to be a factor over the scope $\boldsymbol{Y}' = \boldsymbol{Y} - \boldsymbol{U}$, such that*

$$\phi[\boldsymbol{u}](\boldsymbol{y}') = \phi(\boldsymbol{y}', \boldsymbol{u}) \tag{72}$$

For $\boldsymbol{U} \not\subseteq \boldsymbol{Y}$, define $\phi[\boldsymbol{u}]$ only for the assignments in $\boldsymbol{u}$ to the variables in $\boldsymbol{U}' = \boldsymbol{U} \cap \boldsymbol{Y}$.

Reduced Gibbs distribution:

*Let $P_{\boldsymbol{\Phi}}(\boldsymbol{X})$ be a Gibbs distribution parameterized by $\boldsymbol{\Phi} = \{\phi_1, \ldots, \phi_K\}$ and let $\boldsymbol{u}$ be a context. The **reduced Gibbs distribution** $P_{\boldsymbol{\Phi}}[\boldsymbol{u}]$ is the Gibbs distribution defined by the set of factors $\boldsymbol{\Phi}[\boldsymbol{u}] = \{\phi_1[\boldsymbol{u}], \ldots, \phi_K[\boldsymbol{u}]\}$. More formally:*

$$P_{\boldsymbol{\Phi}}[\boldsymbol{u}] = P_{\boldsymbol{\Phi}}(\boldsymbol{W} \mid \boldsymbol{u}) \quad \textit{where} \quad \boldsymbol{W} = \boldsymbol{X} - \boldsymbol{U} \tag{73}$$

Reduced Markov Network:

*Let $\mathcal{H}$ be a Markov network over $\boldsymbol{X}$ and $\boldsymbol{U} = \boldsymbol{u}$ a context. The **reduced Markov network** $\mathcal{H}[\boldsymbol{u}]$ is a Markov network over the nodes $\boldsymbol{W} = \boldsymbol{X} - \boldsymbol{U}$, where we have an edge X—Y if there's an edge X—Y in $\mathcal{H}$.*

Note that if a Gibbs distribution $P_\Phi(\boldsymbol{X})$ factorizes over $\mathcal{H}$, then $P_\Phi[\boldsymbol{u}]$ factorizes over $\mathcal{H}[\boldsymbol{u}]$.

Markov Network Independencies. A formal presentation of the undirected graph as a representation of independence assertions.

- **Active Path**. Let $\mathcal{H}$ be a Markov network structure, and let $X_1 — \cdots — X_k$ be a path in $\mathcal{H}$. Let $\boldsymbol{Z} \subseteq \mathcal{X}$ be a set of *observed variables*. The path is *active* given $\boldsymbol{Z}$ if none of the X_i's is in $\boldsymbol{Z}$.
- **Separation**. A set of nodes $\boldsymbol{Z}$ *separates* $\boldsymbol{X}$ and $\boldsymbol{Y}$ in $\mathcal{H}$, denoted $\text{sep}_{\mathcal{H}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z})$, if there is no active path between any node $X \in \boldsymbol{X}$ and $Y \in \boldsymbol{Y}$ given $\boldsymbol{Z}$.
- **Global Independencies**. The *global independencies* associated with $\mathcal{H}$ are defined as

$$\mathcal{I}(\mathcal{H}) = \{(\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z}) : \text{sep}_{\mathcal{H}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z})\} \tag{74}$$

  This is the *separation criterion*. Note that the definition of separation is monotonic in $\boldsymbol{Z}$: if it holds for $\boldsymbol{Z}$, then it holds for any $\boldsymbol{Z}' \supset \boldsymbol{Z}$ as well[13].

[13]Which means that Markov networks are fundamentally incapable of representing nonmonotonic independence relations!

- **Soundness** of the separation criterion for detecting independence properties in distributions over $\mathcal{H}$. In other words, we want to prove that

$$(P \text{ factorizes over } \mathcal{H}) \implies [\text{sep}_{\mathcal{H}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z}) \implies P \models (\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})]$$

where the portion in brackets can equivalently be said as "$\mathcal{H}$ is an I-map for P".

Proof

- Consider the case where $\boldsymbol{X} \cup \boldsymbol{Y} \cup \boldsymbol{Z} = \mathcal{X}$.
- Then, any clique in $\mathcal{X}$ is fully contained in either $\boldsymbol{X} \cup \boldsymbol{Z}$ or $\boldsymbol{Y} \cup \boldsymbol{Z}$. In other words,

$$P(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}) = \frac{1}{Z} f(\boldsymbol{X}, \boldsymbol{Z}) g(\boldsymbol{Y}, \boldsymbol{Z}) \tag{75}$$

$$\text{which implies} \quad P(\boldsymbol{X}, \boldsymbol{Y} \mid \boldsymbol{Z}) = \frac{f(\boldsymbol{X}, \boldsymbol{Z}) g(\boldsymbol{Y}, \boldsymbol{Z})}{\sum_{x,y} f(x, \boldsymbol{Z}) g(y, \boldsymbol{Z})} \tag{76}$$

- We can prove that $P \models (\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$ by showing that $P(\boldsymbol{X}, \boldsymbol{Y} \mid \boldsymbol{Z}) = P(\boldsymbol{X} \mid \boldsymbol{Z}) P(\boldsymbol{Y} \mid \boldsymbol{Z})$.

$$\begin{aligned}
P(\boldsymbol{X}, \boldsymbol{Y} \mid \boldsymbol{Z}) &= \frac{f(\boldsymbol{X}, \boldsymbol{Z}) g(\boldsymbol{Y}, \boldsymbol{Z})}{P(\boldsymbol{Z})} && (77) \\
&= \frac{f(\boldsymbol{X}, \boldsymbol{Z}) g(\boldsymbol{Y}, \boldsymbol{Z})}{P(\boldsymbol{Z})} \frac{P(\boldsymbol{Z})}{P(\boldsymbol{Z})} && (78) \\
&= \frac{f(\boldsymbol{X}, \boldsymbol{Z}) g(\boldsymbol{Y}, \boldsymbol{Z})}{P(\boldsymbol{Z})} \frac{\sum_x f(x, \boldsymbol{Z}) \sum_y g(y, \boldsymbol{Z})}{P(\boldsymbol{Z})} && (79) \\
&= \frac{f(\boldsymbol{X}, \boldsymbol{Z}) \sum_y g(y, \boldsymbol{Z})}{P(\boldsymbol{Z})} \frac{g(\boldsymbol{Y}, \boldsymbol{Z}) \sum_x f(x, \boldsymbol{Z})}{P(\boldsymbol{Z})} && (80) \\
&= P(\boldsymbol{X} \mid \boldsymbol{Z}) P(\boldsymbol{Y} \mid \boldsymbol{Z}) && (81)
\end{aligned}$$

- For the general case where $\boldsymbol{X} \cup \boldsymbol{Y} \cup \boldsymbol{Z} = \mathcal{X}$, let $\boldsymbol{U} = \mathcal{X} - (\boldsymbol{X} \cup \boldsymbol{Y} \cup \boldsymbol{Z})$. Since we know that $\text{sep}_{\mathcal{H}}(\boldsymbol{X}; \boldsymbol{Y} \mid \boldsymbol{Z})$, we can partition $\boldsymbol{U}$ into two disjoint sets $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$ such that $\text{sep}_{\mathcal{H}}(\boldsymbol{X} \cup \boldsymbol{U}_1; \boldsymbol{Y} \cup \boldsymbol{U}_2 \mid \boldsymbol{Z})$. Combining the previous result with the decomposition property[a] give us the desired result that $P \models (\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$.

[a]The decomposition property:

$$(\boldsymbol{X} \perp (\boldsymbol{Y}, \boldsymbol{W}) \mid \boldsymbol{Z}) \implies (\boldsymbol{X} \perp \boldsymbol{Y} \mid \boldsymbol{Z})$$

Pairwise Independencies:

Let $\mathcal{H}$ be a Markov network. We define the ***pairwise independencies*** *assoc. with $\mathcal{H}$:*

$$\mathcal{I}_p(\mathcal{H}) = \{(X \perp Y \mid \mathcal{X} - \{X, Y\}) \ : \ X — Y \notin \mathcal{H}\} \tag{82}$$

which just says "X is indep. of Y given everything else if there's no edge between X and Y."

Local Independencies:

> *For a given graph $\mathcal{H}$, define the* ***Markov blanket*** *of X in $\mathcal{H}$, denoted $\mathrm{MB}_{\mathcal{H}}(X)$, to be the neighbors of X in $\mathcal{H}$. We define the* ***local independencies*** *associated with $\mathcal{H}$:*
>
> $$\mathcal{I}_\ell(\mathcal{H}) = \{(X \perp \mathcal{X} - \{X\} - \mathrm{MB}_{\mathcal{H}}(X) \mid \mathrm{MB}_{\mathcal{H}}(X)) \ : \ X \in \mathcal{X}\} \tag{83}$$
>
> which just says "X is indep. of the rest of the nodes in the graph given its immediate neighbors."

Log-Linear Models. Certain patterns involving particular values of variables for a given factor can often be more easily seen by converting factors into log-space. More precisely, we can rewrite a factor $\phi(\boldsymbol{D})$ as

$$\phi(\boldsymbol{D}) = e^{-\epsilon(\boldsymbol{D})} \tag{84}$$

$$\epsilon(\boldsymbol{D}) \triangleq -\ln \phi(\boldsymbol{D}) \tag{85}$$

$$\Pr[X_1, \ldots, X_n] \propto e^{-\sum \epsilon_i(\boldsymbol{D}_i)} \tag{86}$$

where $\epsilon(\boldsymbol{D})$ is often called an **energy function**. Note how ϵ can take on any value along the real line (i.e. removes our nonnegativity constraint)[14]. Also note that as the ϵ summation approaches 0, the probability approaches one.

This motivates introducing the notion of a **feature**, which is just a factor without the nonnegativity requirement. A popular type of feature is the **indicator feature** that takes on value 1 for some values $\boldsymbol{y} \in Val(\boldsymbol{D})$ and 0 otherwise. We can now provide a more general definition for our notion of log-linear models:

> *A distribution P is a* ***log-linear model*** *over a Markov network $\mathcal{H}$ if it is associated with:*
>
> - *A set of features $\mathcal{F} = \{f_1(\boldsymbol{D}_1), \ldots, f_k(\boldsymbol{D}_k)\}$ where each $\boldsymbol{D}_i$ is a complete subgraph (i.e. a clique)in $\mathcal{H}$.*
> - *A set of weights $w_1, \ldots, w_k$.*
>
> *such that*
>
> $$Pr[X_1, \ldots, X_n] = \frac{1}{Z} \exp\left[-\sum_{i=1}^{k} w_i f_i(\boldsymbol{D}_i)\right] \tag{87}$$

The log-linear model provides a much more compact representation for many distributions, especially in situations where variables have large domains (such as text).

[14]We seem to be implicitly assuming that the original factors are all *positive* (not just non-negative).

Box 4.C – Concept: Ising Models and Boltzmann Machines

The **Ising model**: Each atom is modeled as a binary RV $X_i \in \{+1, -1\}$ denoting its spin. Each pair of neighboring atoms is associated with energy function $\epsilon_{i,j}(x_i, x_j) = w_{i,j} x_i x_j$. We also have individual energy functions $u_i x_i$ for each atom. This defines our distribution:

$$P(\xi) = \frac{1}{Z} \exp\left(-\sum_{i<j} w_{i,j} x_i x_j - \sum_i u_i x_i \right) \tag{88}$$

The **Boltzmann distribution**: Now the variables are $X_i \in \{0, 1\}$. The distribution of each X_i given its neighbors is

$$P(x_i^1 \mid Nb(X_i)) = \text{sigmoid}(z) \tag{89}$$

$$z = -(\sum_j w_{i,j} x_j) - u_i \tag{90}$$

Box 4.D – Concept: Metric MRFs

Consider the pairwise graph $X_1, \ldots, X_n$ in the context of sequence labeling. We want to assign each X_i a label. We also want adjacent nodes to prefer being similar to each other. We usually use the MAP objective, so our goal will be to minimize the total energy over the parameters (which are given by the individual energy functions ϵ_i).

$$E(x_1, \ldots, x_n) = \sum_i \epsilon_i(x_i) + \sum_{(i,j)\in\mathcal{E}} \epsilon_{i,j}(x_i, x_j) \tag{91}$$

The simplest place to start for preferring neighboring labels to take on similar values is to define $\epsilon_{i,j}$ to have low energy when $x_i = x_j$ and some positive $\lambda_{i,j}$ otherwise. We want to have finer granularity for our similarities between labels. To do this, we introduce the definition of a **metric**: a function $\mu : \mathcal{V} \times \mathcal{V} \mapsto [0, \infty)$ that satisfies

$$\textbf{[reflexivity]} \quad \mu(v_k, v_l) = 0 \quad \text{IFF} \quad k = l \tag{92}$$

$$\textbf{[symmetry]} \quad \mu(v_k, v_l) = \mu(v_l, v_k) \tag{93}$$

$$\textbf{[triangle inequality]} \quad \mu(v_k, v_l) + \mu(v_l, v_m) \geq \mu(v_k, v_m) \tag{94}$$

and we can now let $\epsilon_{i,j}(v_k, v_l) := \mu(v_k, v_l)$.

Canonical Parameterization (4.4.2.1). Markov networks are generally overparameterized[15]. The **canonical parameterization**, which requires that P be positive, avoids this. First, some notation and requirements:

- P must be positive.
- Let $\xi^* = (x_1^*, \ldots, x_n^*)$ denote some fixed assignment to the network variables $\mathcal{X}$.
- Define $\boldsymbol{x}_{\boldsymbol{Z}} \triangleq \boldsymbol{x}\langle \boldsymbol{Z} \rangle$ as the assignment of variables in some subset $\boldsymbol{Z}$.[16]
- Define $\xi^*_{-\boldsymbol{Z}} \triangleq \xi^*\langle \mathcal{X} - \boldsymbol{Z} \rangle$ be our fixed assignment for the variables outside $\boldsymbol{Z}$.
- Let $\ell(\xi)$ denote $\ln P(\xi)$.

[15] Meaning: for any P_Φ, there are often infinitely many ways to choose its set of parameter values for a given $\mathcal{H}$.

[16] And $\boldsymbol{x}$ is some assignment to some subset of $\mathcal{X}$ that also contains $\boldsymbol{Z}$.

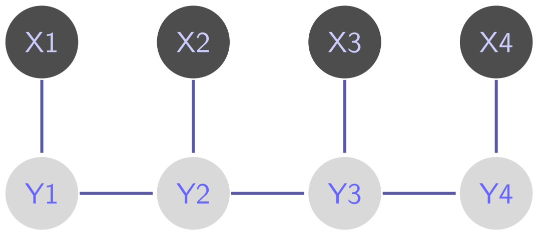
The **canonical energy function** for a clique $\boldsymbol{D}$ is defined below, as well as the associated total $P(\xi)$ over a full network assignment:

The sum is over all subsets of $\boldsymbol{D}$, including $\boldsymbol{D}$ itself and $\varnothing$.

$$\epsilon^*_{\boldsymbol{D}}(\boldsymbol{d}) = \sum_{\boldsymbol{Z} \subseteq \boldsymbol{D}} \ell(\boldsymbol{d}_{\boldsymbol{Z}}, \xi^*_{-\boldsymbol{Z}}) \cdot (-1)^{|\boldsymbol{D}-\boldsymbol{Z}|} \tag{95}$$

$$P(\xi) = \exp\left[\sum_i \epsilon^*_{\boldsymbol{D}_i}(\xi\langle \boldsymbol{D}_i \rangle)\right] \tag{96}$$

Conditional Random Fields. So far, we've only described Markov network representation as encoding a joint distribution over $\mathcal{X}$. The same undirected graph representation and parameterization can also be used to encode a *conditional distribution* $\Pr[\boldsymbol{Y} \mid \boldsymbol{X}]$, where $\boldsymbol{Y}$ is a set of *target variables* and $\boldsymbol{X}$ is a (disjoint) set of *observed variables*.



Note how there are no connection between any of the Xs

Formally, a CRF is an undirected graph $\mathcal{H}$ whose nodes correspond to $\boldsymbol{Y} \cup \boldsymbol{X}$. Since we want to avoid representing[17] a probabilistic model over $\boldsymbol{X}$, we disallow potentials that involve only variables in $\boldsymbol{X}$; our set of factors is $\phi_1(\boldsymbol{D}_1), \ldots, \phi_m(\boldsymbol{D}_m)$, such that each $\boldsymbol{D}_i \nsubseteq \boldsymbol{X}$. The network encodes a conditional distribution as follows:

$$P(\boldsymbol{Y} \mid \boldsymbol{X}) = \frac{1}{Z(\boldsymbol{X})} \widetilde{P}(\boldsymbol{Y}, \boldsymbol{X}) \tag{97}$$

$$\widetilde{P}(\boldsymbol{Y}, \boldsymbol{X}) = \prod_{i=1}^{m} \phi_i(\boldsymbol{D}_i) \tag{98}$$

$$Z(\boldsymbol{X}) = \sum_{\boldsymbol{Y}} \widetilde{P}(\boldsymbol{Y}, \boldsymbol{X}) \tag{99}$$

where now the partition function is a function of the assignment $\boldsymbol{x}$ to $\boldsymbol{X}$.

[17] Also note how we never have to deal with a summation over all possible $\boldsymbol{X}$, due to restricting ourselves to $Z(\boldsymbol{X})$.

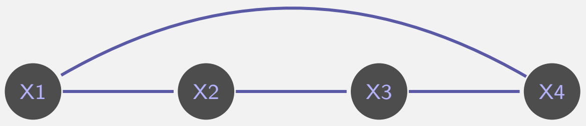
Rapid Summary.

- **Gibbs distribution**: any probability distribution that can be written as a product of factors divided by some partition function Z.
- **Factorizes**: A Gibbs distribution factorizes over $\mathcal{H}$ if each factor [in its product of factors] is a clique.

1.3.1 Exercises

Exercise 4.1

Let $\mathcal{H}$ be the graph of binary variables below. Show that P does not factorize over $\mathcal{H}$ (Hint: proof by contradiction).



- Example 4.4 recap: P satisfies the global independencies w.r.t $\mathcal{H}$. They showed this by manually checking the two global indeps of $\mathcal{H}$, $(X_1 \perp X_3 \mid X_2, X_4)$ and $(X_2 \perp X_4 \mid X_1, X_3)$, against the tabulated list of possible assignments for P (given in example). Nothing fancy.
- P factorizes over $\mathcal{H}$ if it can be written as a product of clique potentials.
- My proof:
  1. Assume that P *does* factorize over $\mathcal{H}$.
  2. Then P can be written as

$$P(X_1, X_2, X_3, X_4) = \frac{1}{Z}\phi_1(X_1, X_2)\phi_2(X_2, X_3)\phi_3(X_3, X_4)\phi_4(X_4, X_1) \tag{100}$$

     Let the above assertion be denoted as C, and the statement that P factorizes according to $\mathcal{H}$ be denoted simply as $P_{\mathcal{H}}$. Since $P_{\mathcal{H}} \iff C$, if we can prove that C does not hold, then we've found our contradiction, meaning $P_{\mathcal{H}}$ also must not hold.
  3. I know that the proof must take advantage of the fact that we know P is zero for certain assignments to $\mathcal{X}$. For example $P(0100) = 0$. Furthermore, by looking at the assignments where P is *not* zero, I can see that all possible combinations of (X_1, X_2) are present, which means ϕ_1 never evaluates to zero.
  4. From the example, we know that $P(1100) = 1/8 \neq 0$. However, since

$$0 = \frac{P(0100)}{P(1100)} = \frac{\phi_1(0,1)\phi_4(0,0)}{\phi_1(1,1)\phi_4(0,1)} \tag{101}$$

$$= \frac{\phi_4(0,0)}{\phi_4(0,1)} \qquad (\phi_1 > 0) \tag{102}$$

     and we also know that both the numerator and denominator of eq. 102 are positive, and thus we have a contradiction.

Exercise 4.4

Prove theorem 4.7 for the case where $\mathcal{H}$ consists of a single clique. Theorem 4.7 is equation 96 in my notes. For a single clique $\boldsymbol{D}$, the question reduces to: Show that, for any assignment $\boldsymbol{d}$ to $\boldsymbol{D}$:

$$P(\boldsymbol{d}) = \frac{\exp\Big(-\epsilon(\boldsymbol{d})\Big)}{\sum_{\boldsymbol{d}'} \exp\Big(-\epsilon(\boldsymbol{d}')\Big)} \tag{103}$$

$$= \exp\Big(\epsilon^*_{\boldsymbol{D}}(\boldsymbol{d})\Big) \tag{104}$$

Consider the case where $|\boldsymbol{D}| = 1$, i.e. $\boldsymbol{d} = d$ is a single variable. Then

$$\epsilon^*_{\boldsymbol{D}}(\boldsymbol{d}) = (-1)^{|\boldsymbol{D}|}\ell(\xi^*_{\boldsymbol{D}}) + (-1)^{|\boldsymbol{D}-\boldsymbol{D}|}\ell(\boldsymbol{d}) \tag{105}$$

$$= -\ell(\xi^*_{\boldsymbol{D}}) + \ell(\boldsymbol{d}) \tag{106}$$

$$= -\ln P(d^*) + \ln P(d) \tag{107}$$

and therefore

$$\exp\Big(\epsilon^*_{\boldsymbol{D}}(\boldsymbol{d})\Big) = P(d)/P(d^*) \tag{108}$$

which is clearly incorrect (???) **TODO**: figure out what's going on here. Either the book has a type in its for theorem 4.7, or I'm absolutely insane.

Probabilistic Graphical Models **May 27, 2018**

Local Probabilistic Models (Ch. 5)

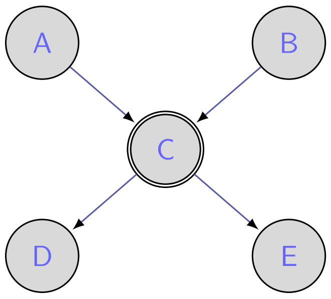
Table of Contents Local *Written by Brandon McKinzie*

Koller and Friedman (2009). Local Probabilistic Models.
Probabilistic Graphical Models: Principles and Techniques.

Deterministic CPDs. When X is a deterministic fu nction of its parents Pa_X:

$$P(x \mid Pa_X) = \begin{cases} 1 & x = f(Pa_X) \\ 0 & \text{otherwise.} \end{cases} \tag{109}$$

Consider the example below, where the double-line notation on C means that C is a deterministic function of A and B. What new conditional dependencies do we have?



Answer: $(D \perp E \mid A, B)$, which would not be true by d-separation alone. It only holds because C is a deterministic function of A and B. z

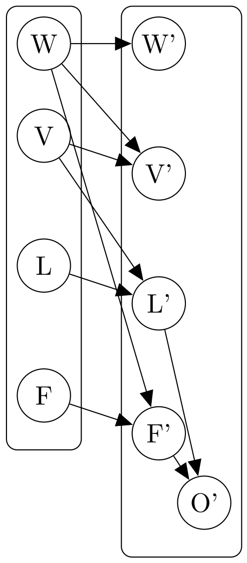
Probabilistic Graphical Models **May 27, 2018**

Template-Based Representations (Ch. 6)

Table of Contents Local *Written by Brandon McKinzie*

Koller and Friedman (2009). Template-Based Representations.
Probabilistic Graphical Models: Principles and Techniques.

In what follows, we will build on an example where a vehicle tries to track its true location (L) using various sensor readings: velocity (V), weather (W), failure of sensor (F), observed location (O) of the noisy sensor.



Temporal Models. We discretize time into slices of interval Δ, and denote the ground random variables at time $t \cdot \Delta$ by $\mathcal{X}^{(t)}$. We can simplify our formulation considerably by assuming a **Markovian system**: a dynamic system over template variables $\mathcal{X}$ that satisfies the Markov assumption:

$$(\mathcal{X}^{(t+1)} \perp \mathcal{X}^{(0:(t-1))} \mid \mathcal{X}^{(t)}) \tag{110}$$

which allows us to define a more compact representation of the joint distribution from time 0 to T:

$$P(\mathcal{X}^{(0:T)}) = P(\mathcal{X}^{(0)}) \prod_{t=0}^{T-1} P(\mathcal{X}^{(t+1)} \mid \mathcal{X}^{(t)}) \tag{111}$$

One last simplifying assumption, to avoid having unique transition probabilities for each time t, is to assume a **stationary**[18] Markovian dynamic system, defined s.t. $P(\mathcal{X}^{(t+1)} \mid \mathcal{X}^{(t)})$ is the same for all t.

[18]Also called time invariant or homogeneous.

Dynamic Bayesian Networks (6.2.2). Above, I've drawn the **2-time-slice Bayesian network** (2-TBN) for our location example. A 2-TBN is a conditional BN over $\mathcal{X}'$ given $\mathcal{X}_I$, where $\mathcal{X}_I \subseteq \mathcal{X}$ is a set of **interface variables**[19]. For each template variable X_i, the CPD $P(X_i' \mid Pa_{X_i'})$ is a **template factor**. We can use the notion of the 2-TBN to define the more general **dynamic Bayesian network**:

> *A **dynamic Bayesian network** (DBN) is a pair $\langle \beta_0, \beta_\to \rangle$, where β_0 is a Bayesian network over $\mathcal{X}^{(0)}$, representing the initial distribution over states, and $\beta_\to$ is a 2-TBN for the process. For any $T \geq 0$, the unrolled Bayesian network is defined such that*
>
> - *$p(X_i^{(0)} \mid Pa_{X_i^{(0)}})$ is the same as the CPD for the corresponding X_i in β_0.*
> - *$p(X_i^{(t)} \mid Pa_{X_i^{(t)}})$ (for $t > 0$) is the same as the CPD for the corresponding X_i' in $\beta_\to$.*

State-Observation Models (6.2.3). Temporal models that, in addition to the Markov assumption (eq. 110), model the observation variables at time t as conditionally independent of the entire state sequence given the variables at time t:

$$\left(\boldsymbol{O}^{(t)} \perp \boldsymbol{X}^{(0:(t-1))}, \boldsymbol{X}^{(t+1:\infty)} \mid \boldsymbol{X}^{(t)} \right) \tag{112}$$

So basically a 2-TBN with the constraint that observation variables are leaves and only have parents in $\boldsymbol{X}'$. We now view our probabilistic model as consisting of 2 components: the *transition model* $P(\boldsymbol{X}' \mid \boldsymbol{X})$, and the *observation model* $P(\boldsymbol{O} \mid \boldsymbol{X})$. The two main architectures for such models are as follows:

- **Hidden Markov Models**. Defined as having a single state variable S and a single observation variable O. In practice, the transition model $P(S' \mid S)$ is often assumed to be sparse (many possible transitions having zero probability). In such cases, one usually represents them visually as **probabilistic finite-state automaton**[20].
- **Linear Dynamical Systems** (LDS) represent a system of one or more real-valued variables that evolve linearly over time, with some Gaussian noise. Such systems are often called **Kalman filters**, after the algorithm used to perform tracking. They can be viewed as a DBN with continuous variables and all dependencies are linear Gaussian[21]. A LDS is traditionally represented as a state-observation model, where both state and observation are vector-valued RVs, and the transition/observation models are encoded using matrices. More formally, for $\boldsymbol{X}^{(t)} \in \mathbb{R}^n, O \in \mathbb{R}^m$:

$$P(\boldsymbol{X}^{(t)} \mid \boldsymbol{X}^{(t-1)}) = \mathcal{N}(A\boldsymbol{X}^{(t-1)}; Q) \tag{113}$$

$Q \in \mathbb{R}^{n \times n}$

$$P(O^{(t)} \mid \boldsymbol{X}^{(t)}) = \mathcal{N}(H\boldsymbol{X}^{(t)}; R) \tag{114}$$

$H \in \mathbb{R}^{m \times m}$

[19]Interface variables are those variables whose values at time t can have a direct effect on the variables at time $t+1$.

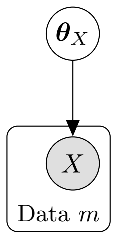
[20]FSA use the graphical notation where the nodes are the individual possible values in $Val(S)$, and the directed edges from some a *to* b have weight equal to $P(S' = b \mid S = a)$.

[21]Linear Gaussian: some var Z pointing to X denotes that $X = \Lambda Z + \text{noise}$, where noise $\sim \mathcal{N}(\mu_x, \Sigma_x)$.

Template Variables and Template Factors (6.3). It's convenient to view the world as being composed of a set of **objects**, which can be divided into a set of mutually exclusive and exhaustive *classes* $\mathcal{Q} = Q_1, \ldots, Q_k$. Template **attributes** have a tuple of *arguments*, each of which is associated with a particular class of objects, which defines the set of objects that can be used to instantiate the argument in a given domain. Template attributes thus provide us with a "generator" for RVs in a given probability space. Formally,

> *An* ***attribute*** *A is a function $A(U_1, \ldots, U_k)$, whose range is some set $Val(A)$, and where each argument U_i is a typed* ***logical variable*** *associated with a particular class $Q[U_i]$. The tuple $U_1, ldots, U_k$ is called the* ***argument signature*** *of the attribute A, and denoted $\alpha(A)$.*

Plate Models (6.4.1). The simplest example of a plate model is shown below. It describes multiple RVs generated from the same distribution $\mathcal{D}$.



This could be a plate model for a set of coin tosses sampled from a single coin. We have a set of m random variables $X(d)$, where $d \in \mathcal{D}$. Each $X(d)$ is the random variable for the dth coin toss. We also explicitly model that the single coin for which the tosses are used is sampled from a distribution θ_X, which takes on values $[0, 1]$ and denotes the bias of the coin.

Probabilistic Graphical Models **July 26, 2018**

Gaussian Network Models (Ch. 7)

Table of Contents Local *Written by Brandon McKinzie*

Koller and Friedman (2009). Gaussian Network Models.
Probabilistic Graphical Models: Principles and Techniques.

Multivariate Gaussians. Here I'll give two forms of the familiar density function, followed by some comments and terminology.

$$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{n/2}\sqrt{\det \Sigma}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right\} \tag{115}$$

$$p(\boldsymbol{x}) \propto \exp\left\{-\frac{1}{2}\boldsymbol{x}^T J \boldsymbol{x} + (J\boldsymbol{\mu})^T \boldsymbol{x}\right\} \quad \text{where } J \triangleq \Sigma^{-1} \tag{116}$$

- The **standard Gaussian** is defined as $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$.
- Σ must be *positive definite*[22]: $\forall \boldsymbol{x} \neq 0,\ \boldsymbol{x}^T \Sigma \boldsymbol{x} > 0$. Recall that $\Sigma_{i,j} = \mathrm{Cov}\left[x_i, x_j\right] = \mathbb{E}\left[x_i x_j\right] - \mu_i \mu_j$.

The two operations we usually want to perform on a Gaussian are (1) computing marginals, and (2) conditioning the distribution on an assignment of some subset of the variables. For (1), its easier to use the standard form of $p(\boldsymbol{x})$, whereas for (2) it is easier to use the information form (the one using J).

Multivariate Gaussians are also special because we can easily determine whether two x_i and x_j are independent: $x_i \perp x_j$ IFF $\Sigma_{i,j} = 0$[23]. For conditional independencies, the information matrix J is easier to work with: $(x_i \perp x_j \mid \{x\}_{k \notin \{i,j\}})$ IFF $J_{i,j} = 0$. This condition is also how we defined pairwise independencies in a Markov network, which leads to the awesome realization:

> *We can view the information matrix J as directly defining a minimal I-map Markov network for [multivariate Gaussian] p, whereby any entry $J_{i,j} \neq 0$ corresponds to an edge x_i—x_j in the network.*

[22]Most definitions of p.d. also require that the matrix be symmetric. I also think it helps to view positive definite from the operator perspective: *A linear operator T is positive definite if T is self-adjoint and $\langle T(x), x\rangle > 0$. In other words, "positive definite" means that the result of applying the matrix/operator to any nonzero $\boldsymbol{x}$ will always have a positive component along the original direction $\hat{\boldsymbol{x}}$.*

[23]This is not true in general – just for multivariate Gaussians!

Probabilistic Graphical Models **May 06, 2018**

Variable Elimination (Ch. 9)

Table of Contents Local *Written by Brandon McKinzie*

Koller and Friedman (2009). Variable Elimination.
Probabilistic Graphical Models: Principles and Techniques.

Analysis of Exact Inference. The focus of this chapter is the conditional probability query,

$$\Pr[\boldsymbol{Y} \mid \boldsymbol{E} = \boldsymbol{e}] = \frac{\Pr[\boldsymbol{Y}, \boldsymbol{e}]}{\Pr[\boldsymbol{e}]} \tag{9.1}$$

Ideally, we want to obtain all instantiations $\Pr[\boldsymbol{y} \mid \boldsymbol{e}]$ of equation 9.1. Let $\boldsymbol{W} = \mathcal{X} - \boldsymbol{Y} - \boldsymbol{E}$ be the RVs that are neither query nor evidence. Then

$$\Pr[\boldsymbol{y} \mid \boldsymbol{e}] = \frac{\sum_{\boldsymbol{w}} \Pr[\boldsymbol{y}, \boldsymbol{e}, \boldsymbol{w}]}{\sum_{\boldsymbol{y}} \Pr[\boldsymbol{y}, \boldsymbol{e}]} \tag{9.3}$$

and note that, by computing all instantiations for the numerator first, we can reuse them to obtain the denominator. We can formulate the inference problem as a decision problem, which we will call $BNPrDP$, defined as follows:

Given: Bayesian network $\mathcal{B}$ over $\mathcal{X}$, a variable $X \in \mathcal{X}$, and a value $x \in Val(X)$.
Decide: whether $Pr_{\mathcal{B}}[X = x] > 0$.

$BNPrDP$

Thm 9.1: The decision problem $BNPrDP$ is $\mathcal{NP}$-complete. **Proof**:

1. **BNPrDP is in $\mathcal{NP}$**: Guess a full assignment ξ to the network[24]. If the guess is successful, where sucess if defined as $(X = x) \in \xi$ and $P(\xi) > 0$, then we know that $P(X = x) > 0$[25]. Computing $P(\xi)$ is linear in the number of factors for a BN, since we just multiply them together.
2. **BNPrDP is $\mathcal{NP}$-hard**. We show this by proving that we can solve 3-SAT (which is $\mathcal{NP}$-hard) by transforming inputs to 3-SAT to inputs of BNPrDP in polynomial time. Given any 3-SAT formula ϕ, we can create a BN B_ϕ with some special variable X s.t. ϕ is satisfiable IFF $P_{B_\phi}(X = x^1) > 0$. You can easily build such a network by having a node Q_i for each binary RV q_i, and a node C_i for each of the clauses that's a deterministic function of its parents (up to 3 Q nodes). Then, the node X is a deterministic function of its parents, which are chains of AND gates along the C_i. Since each node has at most 3 parents, we can ensure that construction is bounded by polynomial time in the length of ϕ.

[24] Apparently the time it takes to generate a guess is irrelevant.
[25] This is true because $P(x)$ can be decomposed as $P(\xi) + \sum P(\ldots) \geq P(\xi)$.

Analysis of Approximate Inference. Consider a specific query $P(\boldsymbol{y} \mid \boldsymbol{e})$, where we focus on a particular assignment $\boldsymbol{y}$. Let ρ denote some approximate answer, whose accuracy we wish to evaluate relative to the correct probability. We can use the **relative error** to estimate the quality of the approximation: *An estimate ρ has relative error ϵ if:*

$$\frac{\rho}{1+\epsilon} \leq P(\boldsymbol{y} \mid \boldsymbol{e}) \leq \rho(1+\epsilon) \tag{117}$$

Unfortunately, the task of finding some approximation ρ with relative error ϵ *is also $\mathcal{NP}$-hard*. Furthermore, even if we relax this metric by using absolute error instead, we end up finding that **in the case where we have evidence, approximate inference is no easier than exact inference, in the worst case**.

Variable Elimination. Basically, a dynamic programming approach for performing exact inference. Consider the simple BN $X_1 \to \cdots \to X_n$, where each variable can take on k possible values. The dynamic programming approach for computing $P(X_n)$ involves computing

$$P(X_{i+1}) = \sum_{x_i} P(X_{i+1} \mid x_i) P(x_i) \tag{118}$$

$n-1$ times, starting with $i=1$, all the way up to $i=n-1$, reusing the previous computation at each step, with total cost $\mathcal{O}(nk^2)$. So for this simple network, even though the size of the joint is k^n (exponential in n), we can do inference in linear time.

First, we formalize some basic concepts before defining the algorithm.

Factor marginalization:

> *Let $\boldsymbol{X}$ be a set of variables, and $Y \notin \boldsymbol{X}$ a variable. Let $\phi(\boldsymbol{X}, Y)$ be a factor. Define the **factor marginalization** of Y in ϕ, denoted $\sum_Y \phi$, to be a factor ψ over $\boldsymbol{X}$ such that:*
>
> $$\psi(\boldsymbol{X}) = \sum_Y \phi(\boldsymbol{X}, Y)$$

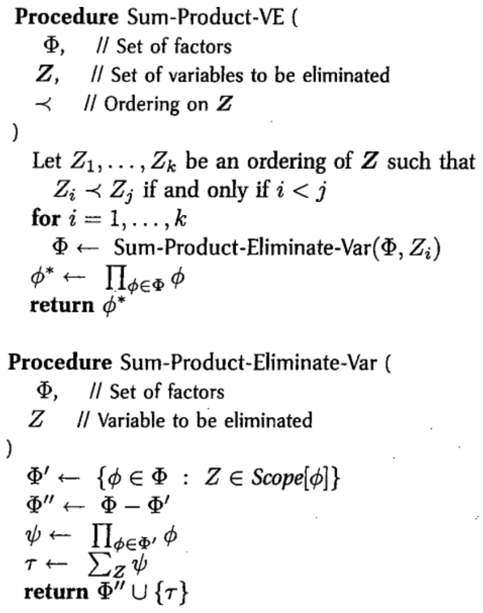
The key observation that's easy to miss is that *we're only summing entries in the table where the values of $\boldsymbol{X}$ match up.* One useful rule for exchanging factor product and summation: If $X \notin Scope[\phi_1]$, then

$$\sum_X (\phi_1 \cdot \phi_2) = \phi_1 \cdot \sum_X \phi_2 \tag{9.6}$$

So, when computing some marginal probability, the main idea is to group factors together and compute expressions of the form

$$\sum_{\boldsymbol{Z}} \prod_{\phi \in \Phi} \phi \tag{119}$$

where Φ is the set of all factors ϕ for which $\boldsymbol{Z} \in Scope[\phi]$. This is commonly called the **sum-product** inference task. The full algorithm for sum-product variable elimination, which is an instantiation of the sum-product inference task, is illustrated below.

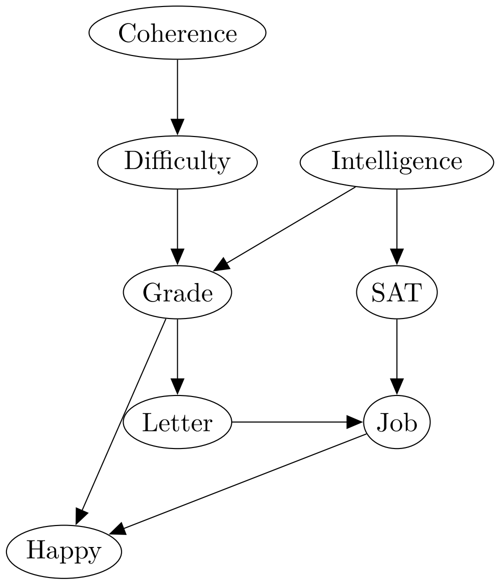


```
Procedure Sum-Product-VE (
    $\Phi$,    // Set of factors
    $\boldsymbol{Z}$,    // Set of variables to be eliminated
    $\prec$    // Ordering on $\boldsymbol{Z}$
)
  Let $Z_1, \ldots, Z_k$ be an ordering of $\boldsymbol{Z}$ such that
    $Z_i \prec Z_j$ if and only if $i < j$
  for $i = 1, \ldots, k$
    $\Phi \leftarrow$ Sum-Product-Eliminate-Var($\Phi, Z_i$)
  $\phi^* \leftarrow \prod_{\phi \in \Phi} \phi$
  return $\phi^*$

Procedure Sum-Product-Eliminate-Var (
    $\Phi$,    // Set of factors
    $Z$    // Variable to be eliminated
)
  $\Phi' \leftarrow \{\phi \in \Phi : Z \in Scope[\phi]\}$
  $\Phi'' \leftarrow \Phi - \Phi'$
  $\psi \leftarrow \prod_{\phi \in \Phi'} \phi$
  $\tau \leftarrow \sum_{Z} \psi$
  return $\Phi'' \cup \{\tau\}$
```

This is what we use to compute the marginal probability $P(\boldsymbol{X})$ where $\boldsymbol{X} = \mathcal{X} - \boldsymbol{Z}$. To compute conditional queries of the form $P(\boldsymbol{Y} \mid \boldsymbol{E} = \boldsymbol{e})$, simply replace all factors whose scope overlaps with $\boldsymbol{E}$ with their reduced factor (see chapter 4 notes for definition) to get the unnormalized $\phi^*(\boldsymbol{Y})$ (the numerator of $P(\boldsymbol{Y} \mid \boldsymbol{e})$). Then divide by $\sum_{\boldsymbol{y}} \phi^*$ to obtain the final result.

Example. We will work through computing $P(Job)$ for the BN below.



Due to Happy being a child of Job, $P(J)$ actually requires using all factors in the graph. Below shows how VE with elimination ordering C, D, I, H, G, S, L progressively simplifies the equation for computing $P(J)$.

$$\begin{aligned}
P(J) &= \sum P(J \mid s, \ell)P(s \mid i)P(\ell \mid g)P(g \mid d, i)P(d \mid c)P(h \mid J, g)P(c)P(i) && (120)\\
&= \sum_{c,d,i,h,g,s,\ell} (P(c)P(d \mid c)) \cdot P(g \mid d, i)P(i)P(s \mid i)P(h \mid J, g)P(\ell \mid g)P(J \mid s, \ell) && (121)\\
&= \sum_{d,i,h,g,s,\ell} (\tau_1(d)P(g \mid d, i)) \cdot P(i)P(s \mid i)P(h \mid J, g)P(\ell \mid g)P(J \mid s, \ell) && (122)\\
&= \sum_{i,h,g,s,\ell} (\tau_2(g, i)P(i)P(s \mid i)) \cdot P(h \mid J, g)P(\ell \mid g)P(J \mid s, \ell) && (123)\\
&= \sum_{h,g,s,\ell} (P(h \mid J, g)) \cdot \tau_3(g, s)P(\ell \mid g)P(J \mid s, \ell) && (124)\\
&= \sum_{g,s,\ell} (\tau_4(g, J)\tau_3(g, s)P(\ell \mid g)) \cdot P(J \mid s, \ell) && (125)\\
&= \sum_{s,\ell} \tau_5(J, \ell, s) \cdot P(J \mid s, \ell) && (126)\\
&= \sum_{\ell} \tau_6(J, \ell) && (127)
\end{aligned}$$

where red indicates the focus of the given step in the VE algorithm.

Probabilistic Graphical Models **May 06, 2018**

Clique Trees (Ch. 10)

Table of Contents Local *Written by Brandon McKinzie*

Koller and Friedman (2009). Clique Trees.
Probabilistic Graphical Models: Principles and Techniques.

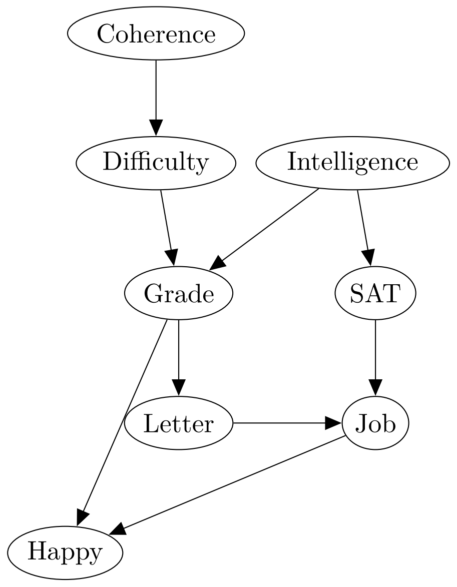
Cluster Graphs. A graphical flowchart of the factor-manipulation process that will be relevant when we discuss message passing. Each node is a *cluster*, which is associated with a subset of variables. Formally,

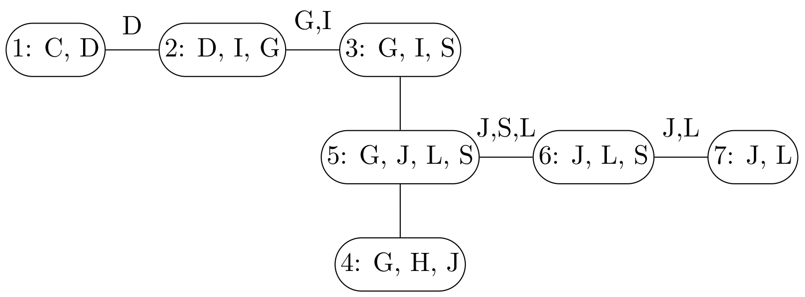
> *A* ***cluster graph*** $\mathcal{U}$ *for a set of factors* Φ *over* $\mathcal{X}$ *is an undirected graph. Each node* i *is associated with a subset* $\boldsymbol{C}_i \subseteq \mathcal{X}$*. Each factor* $\phi \in \Phi$ *must be associated with a cluster* $\boldsymbol{C}_i$*, denoted* $\alpha(\phi)$*, such that* $Scope[\phi] \subseteq \boldsymbol{C}_i$ *(**family-preserving**). Each edge between a pair of clusters* $\boldsymbol{C}_i$ *and* $\boldsymbol{C}_j$ *is associated with a* ***sepset*** $\boldsymbol{S}_{i,j} \subseteq \boldsymbol{C}_i \cap \boldsymbol{C}_j$*.*

Recall that each step of variable elimination involves creating a factor ψ_i by multiplying a group of factors[26]. Then, denoting the variable we are eliminating at this step as Z, we obtain another factor τ_i that's the factor marginalization of Z in ψ_i (denoted $\sum_Z \psi_i$). An execution of variable elimination defines a cluster graph: we have a cluster for each of the ψ_i, defined as $\boldsymbol{C}_i = Scope[\psi_i]$. We draw an edge between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$ if the **message** τ_i is used in the computation of τ_j.

Consider when we applied variable elimination to the student graph network below, to compute $P(J)$. Elimination ordering C, D, I, H, G, S, L.

[26] All whose scope contains the variable we are currently trying to eliminate.





Clique Trees. Since VE uses each intermediate τ_i at most once, the cluster graph induced by an execution VE is necessarily a *tree*, and it also defines a directionality: the direction of the message passing (left-to-right in the above illustration). All the messages flow toward a single cluster where the final result is computed – the **root** of the tree; we say the messages "flow up" to the root. Furthermore, for cluster trees induced by VE, the scope of each message (edge) τ_i is *exactly* $\boldsymbol{C}_i \cap \boldsymbol{C}_j$, not just a subset[27].

> *Let Φ be a set of factors over $\mathcal{X}$. A cluster tree over Φ that satisfies the running intersection property is called a* ***clique tree****. For clique trees, the clusters are also called* ***cliques****.*

Message Passing: Sum Product. Previously we saw how an execution of VE can be illustrated with a clique tree. We now go the other direction – given a clique tree, we show how it can be used for variable elimination. Given a clique tree representation of some BN, we can use it to guide us along an execution of VE to compute any marginal we'd like. First, before any run, we generate the set of **initial potentials** ψ_i associated with each clique $\boldsymbol{C}_i$ in the tree, defined as just the multiplication of the initial factors associated with the clique. We define the root of the tree as any clique containing the variable whose marginal we want to compute (we pick arbitrarily). Starting from the leaves and moving toward the root, we pass messages along from clique to clique. A clique is *ready* to send a message when it has received a message from all of its downstream neighbors. The message from $\boldsymbol{C}_i$ to [a neighbor] $\boldsymbol{C}_j$ is computed using the **sum-product message passing** computation:

[27]This follows from the **running intersection property**, which is satisfied by any cluster tree that's defined by variable elimination. It's defined as, if any variable X is in both cluster $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$, then X is also in every cluster in the (unique) path in the tree between $\boldsymbol{C}_i$ and $\boldsymbol{C}_j$.

$$\delta_{i\to j} = \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi_i \cdot \prod_{k\in(Nb_i - \{j\})} \delta_{k\to i} \tag{10.2}$$

where the summation is simply over the variables in $\boldsymbol{C}_i$ that aren't passed along to $\boldsymbol{C}_j$, and the product is over all messages that $\boldsymbol{C}_i$ received. Stated even simpler, we multiply all the incoming messages by our initial potential, then sum out all variables except those in $\boldsymbol{S}_{i,j}$. When the root clique has received all messages, it multiplies them with its own initial potential, resulting in a factor called the **beliefs**, $\beta_r(\boldsymbol{C}_r)$. It represents

$$\widetilde{P}_\Phi(\boldsymbol{C}_r) = \sum_{\mathcal{X}-\boldsymbol{C}_r} \prod_\phi \phi \tag{128}$$

where, to be clear, the product is over all ϕ in the graph.

Below is a more compact summary of all of this, showing the procedure for computing *all* final factors (belief) β_i for some marginal probability query on the variables in $\boldsymbol{C}_r$ *asynchronously.*
Algorithm 10.2: Sum-Product Belief Propagation

1. For each clique $\boldsymbol{C}_i$, compute its initial potential:
$$\psi_i(\boldsymbol{C}_i) \leftarrow \prod_{\phi_j:\alpha(\phi_j)=i} [\phi_j]$$
2. While $\exists i, j$ such that i is ready to transmit to j, compute:
$$\delta_{i\to j} \leftarrow \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \psi_i \cdot \prod_{k\in(Nb_i - \{j\})} \delta_{k\to i}$$
3. Then, compute each belief factors β_i by multiplying the initial potential ψ_i by the incoming messages to $\boldsymbol{C}_i$:
$$\beta_i \leftarrow \psi_i \cdot \prod_{k\in Nb_{C_i}} \delta_{k\to i}$$
4. Return the set of beliefs $\{\beta_i\}$, where
$$\beta_i = \sum_{\mathcal{X}-\boldsymbol{C}_i} \widetilde{P}_\Phi(\mathcal{X}) = \widetilde{P}_\Phi(\boldsymbol{C}_i) \tag{129}$$

The SP Belief Propagation algorithm above is also called **clique tree calibration**. A clique tree $\mathcal{T}$ is *calibrated* if all pairs of adjacent cliques are calibrated. A calibrated clique tree satisfies the following property for what we'll now call the **clique beliefs**, β_i, and the **sepset beliefs**, $\mu_{i,j}$ over $\boldsymbol{S}_{i,j}$:

$$\mu_{i,j}(\boldsymbol{S}_{i,j}) \triangleq \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \beta_i = \sum_{\boldsymbol{C}_j - \boldsymbol{S}_{i,j}} \beta_j \tag{130}$$

$\mu_{i,j} = \widetilde{P}_\Phi(\boldsymbol{S}_{i,j})$

The main advantage of the clique tree algorithm is that it computes the posterior probability of all variables in a graphical model using only twice the computation[28] *of the upward pass in the same tree.*

[28]The algorithm is equivalent to doing one upward pass, one downward pass.

We can also show that $\mu_{i,j} = \delta_{j\to i}\delta_{i\to j}$, which then allows us to derive:

$$\widetilde{P}_\Phi(\mathcal{X}) = \frac{\prod_{i\in\mathcal{V}_\mathcal{T}} \beta_i}{\prod_{ij\in\mathcal{E}_\mathcal{T}} \mu_{i,j}} \tag{10.10}$$

In other words, the clique and sepset beliefs provide a **reparameterization** of the unnormalized measure, a property called the **clique tree invariant**.

Message Passing: Belief Update. We know discuss an alternative message passing approach that is mathematically equivalent but intuitively different than the sum-product approach. First, we introduce some new definitions.

Factor Division:

Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be disjoint sets of variables, and let $\phi_1(\boldsymbol{X},\boldsymbol{Y})$ and $\phi_2(\boldsymbol{Y})$ be two factors. We define the division of ϕ_1 and ϕ_2 as a factor ψ with scope $\boldsymbol{X},\boldsymbol{Y}$ as follows:

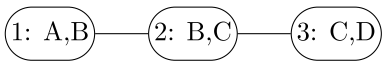
Define $0/0 = 0$

$$\psi(\boldsymbol{X},\boldsymbol{Y}) \triangleq \frac{\phi_1(\boldsymbol{X},\boldsymbol{Y})}{\phi_2(\boldsymbol{Y})}$$

Looking back at equation 10.2, we can now see that another way to write $\delta_{i\to j}$ is

$$\delta_{i\to j} = \frac{\sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \beta_i}{\delta_{j\to i}} \tag{10.13}$$

Now, consider the clique tree below for the simple Markov network A-B-C-D:



If we assigned $\boldsymbol{C}_2$ as the root, then our previous approach would compute $\delta_{2\to 1}$ as $\sum_C \psi_2 \cdot \delta_{3\to 2}$. Alternatively, we can use equation 10.13 to realize this is equivalent to dividing β_2 by $\delta_{1\to 2}$ and marginalizing out C. This observation motivates the algorithm below, which allows us to execute message passing in terms of the clique and sepset beliefs, without having to remember the initial potentials ψ_i or explicitly compute the messages $\delta_{i\to j}$.

Algorithm 10.3: Belief-Update Message Passing

1. For each clique $\boldsymbol{C}_i$, set its initial belief β_i to its initial potential ψ_i. For each edge in $\mathcal{E}_\mathcal{T}$, set $\mu_{i,j} = \mathbf{1}$.
2. Wile there exists an uninformed[29] clique in $\mathcal{T}$, select any edge in $\mathcal{E}_\mathcal{T}$, and compute

[29] A clique is informed once it has received informed messages from all of its neighbors. An informed message is one that has been sent by taking into account information from all of the sending cliques' neighbors (aside from the receiving clique of that message, of course).

$$\sigma_{i \to j} \leftarrow \sum_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} [\beta_i] \tag{131}$$

$$\beta_j \leftarrow \beta_j \cdot \frac{\sigma_{i \to j}}{\mu_{i,j}} \tag{132}$$

$$\mu_{i,j} \leftarrow \sigma_{i \to j} \tag{133}$$

3. Return the resulting set of informed beliefs $\{\beta_i\}$.

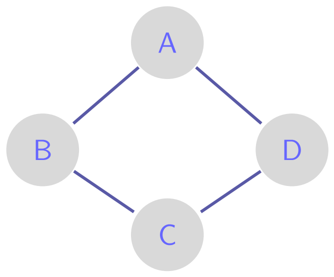
At convergence, $\sigma_{i \to j} = \mu_{i,j} = \sigma_{j \to i}$.

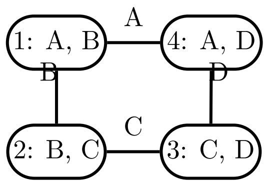
Inference as Optimization (Ch. 11)

Written by Brandon McKinzie

Koller and Friedman (2009). Inference as Optimization.
Probabilistic Graphical Models: Principles and Techniques.

Propagation-Based Approximation. We can use a general-purpose cluster graph rather than the more restrictive clique tree (needed to guarantee exact inference) for approximate inference methods. Consider the simple Markov network below on the left.





The clique tree for this network, which can be used for exact inference, has two cliques ABD and BCD and messages are passed between them consisting of $\tau(B, D)$. Suppose that, instead, we set up 4 clusters corresponding to each of the initial potentials, shown as the cluster graph above on the right. We can still apply belief propagation here, but due to it now having loops (as opposed to before when we only had trees), *the process may not converge.*

Probabilistic Graphical Models **June 17, 2018**

Parameter Estimation (Ch. 17)

Table of Contents Local *Written by Brandon McKinzie*

Koller and Friedman (2009). Parameter Estimation.
Probabilistic Graphical Models: Principles and Techniques.

Maximum Likelihood Estimation (17.1). In this chapter, assume the network structure is fixed and that our data set $\mathcal{D}$ consists of fully observed instances of the network variables: $\mathcal{D} = \{\xi[1], \ldots, \xi[M]\}$. We begin with the simplest learning problem: parameter learning for a single variable. We want to estimate the probability, denoted via the *parameter* θ, with which the flip of a thumbtack will land heads or tails. Define the **likelihood function** $L(\theta : \boldsymbol{x})$ as the probability of observing some sequence of outcomes $\boldsymbol{x}$ under the parameter θ. In other words, it is simply $P(\boldsymbol{x} : \theta)$, but interpreted as a *function of* θ. For our simple case, where $\mathcal{D}$ consists of M thumbtack flip outcomes,

$$L(\theta : \mathcal{D}) = \theta^{M[1]}(1-\theta)^{M[0]} \tag{134}$$

where $M[1]$ denotes the number of outcomes in $\mathcal{D}$ that were heads. Since it's easier to maximize a logarithm, and since it yields the same optimal $\hat{\theta}$, optimize the **log-likelihood** to obtain:

$$\begin{aligned}\hat{\theta} &= \arg\max_\theta \ell(\theta : \mathcal{D}) = \arg\max_\theta \left[M[1]\log\theta + M[0]\log(1-\theta)\right] && (135)\\ &= \frac{M[1]}{M[1]+M[0]} && (136)\end{aligned}$$

Note that MLE has the *disadvantage* that it can't communicate confidence of an estimate[30].

We now provide the more general formal definitions for MLE.

- We are given a *training set* $\mathcal{D}$ containing M (IID) instances of a set of random variables $\mathcal{X}$, where the samples of $\mathcal{X}$ are drawn from some unknown distribution $P^*(\mathcal{X})$.
- We are given a **parametric model**, defined by a function $P(\xi; \boldsymbol{\theta})$, where ξ is an instance of $\mathcal{X}$, and we want to estimate its parameters $\boldsymbol{\theta}$[31]. The model also defines the space of legal parameter values Θ, the **parameter space**.
- We then define the **likelihood function** $L(\boldsymbol{\theta} : \mathcal{D}) = \prod_m P(\xi[m] : \boldsymbol{\theta})$.

[30]We get the same result (0.3) if we get 3 heads out of 10 flips, as we do for getting 300 heads out of 1000 flips; yet, the latter experiment should include a higher degree of confidence

[31]We also have the constraint that $P(\xi; \boldsymbol{\theta})$ must be a valid distribution (nonnegative and sums to 1 over all possible ξ)

We can often simplify the likelihood function to simpler terms, like our $M[0]$ and $M[1]$ values in the thumbtack example. These are called the **sufficient statistics**, defined as functions of the data that summarize the relevant information for computing the likelihood. Formally,

A function $\tau(\xi) : \xi \to \mathbb{R}^\ell$ (for some ℓ) is a ***sufficient statistic*** *if for any two data sets $\mathcal{D}$ and $\mathcal{D}'$, we have that*

$$\left[\sum_{\xi[m]\in\mathcal{D}} \tau(\xi[m]) = \sum_{\xi'[m]\in\mathcal{D}'} \tau(\xi'[m])\right] \Longrightarrow \left[L(\boldsymbol{\theta} : \mathcal{D}) = L(\boldsymbol{\theta} : \mathcal{D}')\right] \tag{137}$$

We often informally refer to the tuple $\sum_{\xi[m]\in mathcalD} \tau(\xi[m])$ as the sufficient statistics of the data set $\mathcal{D}$.

MLE for Bayesian Networks – Simple Example. We now move on to estimating parameters $\boldsymbol{\theta}$ for the simple BN $X \to Y$ for two binary RVs X and Y. Our parameters $\boldsymbol{\theta}$ are the individual probabilities of $P(X)$ and $P(Y \mid X)$ (6 total). Since BNs have the nice property that their joint probability decomposes into a product of probabilities, just like how the likelihood function is a product of probabilities, we can write the likelihood function as a product of the individual *local* probabilities:

decomposability of the likelihood function

$$L(\boldsymbol{\theta} : \mathcal{D}) = \left(\prod_m P(x[m] : \boldsymbol{\theta}_X)\right)\left(\prod_m P(y[m] \mid x[m] : \boldsymbol{\theta}_{Y|X})\right) \tag{138}$$

which can be decomposed even further by e.g. differentiating products over $x[m] : x[m] = x^0$ etc. Just as we used $M[0]$ in the thumbtack example to count the number of instances with a certain value, we can use the same idea for the general case.

Let $\boldsymbol{Z}$ be some set of RVs, and $\boldsymbol{z}$ be some instantiation to them. We define $M[\boldsymbol{z}]$ to be the number of entries in data set $\mathcal{D}$ that have $\boldsymbol{Z}[m] = \boldsymbol{z}$:

$$M[\boldsymbol{z}] = \sum_m \mathbb{1}\{\boldsymbol{Z}[m] = \boldsymbol{z}\} \tag{139}$$

Global Likelihood Decomposition. We now move to the more general case of computing the likelihood for BN with structure $\mathcal{G}$.

global decomposition of the likelihood

$$\begin{aligned}
L(\boldsymbol{\theta} : \mathcal{D}) &= \prod_m P_{\mathcal{G}}(\xi[m] : \boldsymbol{\theta}) && (140)\\
&= \prod_m \prod_i P(x_i[m] \mid Pa_{X_i}[m] : \boldsymbol{\theta}) && (141)\\
&= \prod_i \left[\prod_m P(x_i[m] \mid Pa_{X_i}[m] : \boldsymbol{\theta}_{X_i|Pa_{X_i}})\right] && (142)\\
&= \prod_i L_i(\boldsymbol{\theta}_{X_i|Pa_{X_i}} : \mathcal{D}) && (143)
\end{aligned}$$

where L_i is the **local likelihood function** for X_i. Assuming these are each disjoint sets of parameters from one another, it implies that $\hat{\boldsymbol{\theta}} = \langle \hat{\boldsymbol{\theta}}_{X_1|Pa_{X_1}}, \ldots, \hat{\boldsymbol{\theta}}_{X_n|Pa_{X_n}} \rangle$

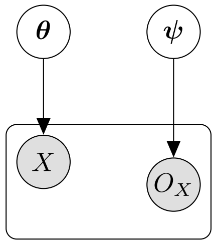
Probabilistic Graphical Models **June 17, 2018**

Partially Observed Data (Ch. 19)

Table of Contents Local *Written by Brandon McKinzie*

Koller and Friedman (2009). Partially Observed Data.
Probabilistic Graphical Models: Principles and Techniques.

Likelihood of Data and Observation Models (19.1.1). Consider the simple example of flipping a thumbtack, but occasionally the thumbtack rolls off the table. We choose to ignore the tosses for which the thumbtack rolls off. Now, in addition to the random variable X giving the flip outcome, we have the *observation variable* O_X, which tells us whether we observed the value of X.



The illustration above is a plate model where we choose a thumbtack sampled with bias $\boldsymbol{\theta}$ and repeat some number of flips with that same thumbtack. We also sample the random variable O_X that has probability of observation sampled from ψ and fixed for all the experiments we do. This leads to the following definition for the observability model.

> *Let $\boldsymbol{X} = \{X_1, \dots X_n\}$ be some set of RVs, and let $O_{\boldsymbol{X}} = \{O_{X_1}, \dots, O_{X_n}\}$ be their* ***observability variable****. The* ***observability model*** *is a join distribution*
>
> $$P_{missing}(\boldsymbol{X}, O_{\boldsymbol{X}}) = P(\boldsymbol{X}) \cdot P_{missing}(O_{\boldsymbol{X}} \mid \boldsymbol{X})$$
>
> *so that $P(\boldsymbol{X})$ is parameterized by $\boldsymbol{\theta}$ and $P_{missing}(O_{\boldsymbol{X}} \mid \boldsymbol{X})$ is parameterized by $\boldsymbol{\psi}$. We define a new set of RVs $\boldsymbol{Y} = \{Y_1, \dots Y_n\}$ where $Val(Y_i) = Val(X_i) \cup \{?\}$. The actual observation $\boldsymbol{Y}$ is a deterministic function of $\boldsymbol{X}$ and $O_{\boldsymbol{X}}$:*
>
> $$Y_i = \begin{cases} X_i & O_{X_i} = o^1 \\ ? & O_{X_i} = o^0 \end{cases} \tag{144}$$

For our simple model above, we have

$$P(Y = 1) = \theta\psi \tag{145}$$
$$P(Y = 0) = (1 - \theta)\psi \tag{146}$$
$$P(Y = ?) = (1 - \psi) \tag{147}$$
$$L(\theta, \psi; \mathcal{D}) = \theta^{M[1]}(1 - \theta)^{M[0]}\psi^{M[1]+M[0]}(1 - \psi)^{M[?]} \tag{148}$$

The main takeaway is to understand that **when we have missing data, the data-generation process involves two steps: (1) generate data by sampling from the model, then (2) determine which values we get to observe and which ones are hidden from us**.

The Likelihood Function (19.1.3). Assume we have a BN network $\mathcal{G}$ over a set of variables $\boldsymbol{X}$. In general, each instance has a different set of observed variables. Denote by $\boldsymbol{O}[m]$ and $\boldsymbol{o}[m]$ the observed vars and their values in the m'th instance, and by $\boldsymbol{H}[m]$ the missing (or hidden) vars in the m'th instance.

Information Theory, Inference, and Learning Algorithms

Contents

Information Theory, Inference, and Learning Algorithms **November 11, 2017**

Introduction to Information Theory (Ch. 1)

Table of Contents Local *Written by Brandon McKinzie*

[Note: Skipping most of this chapter since it's mostly introductory material.]

Preface. For ease of reference, some common quantities we will frequently be using:

- **Binomial distribution**. Let r denote the number of successful trials out of N total trials. Let f denote the probability of success for a single trial.

$$\Pr\left[r \mid f, N\right] = \binom{N}{r} f^r (1-f)^{N-r} \qquad \mathbb{E}\left[r\right] = Nf \qquad \mathrm{Var}\left[r\right] = Nf(1-f) \tag{149}$$

- **Stirling's Approximation**.

$$x! \simeq x^x e^{-x}\sqrt{2\pi x} \quad \Leftrightarrow \quad \ln x! \simeq x \ln x - x + \frac{1}{2}\ln 2\pi x \tag{150}$$

Recall that $\log_b x = \frac{\log_a x}{\log_a b}$

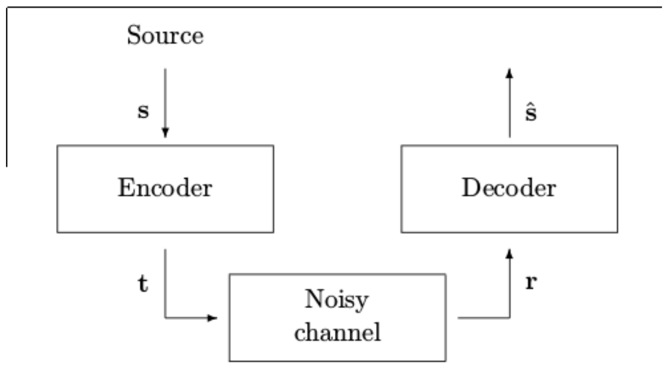
$$\ln \binom{N}{r} \simeq r \ln \frac{N}{r} + (N-r)\ln\frac{N}{N-r} \tag{151}$$

- **Binary Entropy Function** and its relation with Stirling's approximation.

$$H_2(x) \triangleq x \lg \frac{1}{x} + (1-x)\lg\frac{1}{1-x} \tag{152}$$

$$\lg \binom{N}{r} \simeq N H_2(r/N) \tag{153}$$

Perfect communication over an imperfect, noisy communication channel (1.1). We want to make an encoder-decoder architecture, of the general form in the figure below, to achieve reliable communication over a noisy channel.



Information theory is concerned with the theoretical limitations and potentials of such systems. Let's explore some examples for the case of the **binary symmetric channel**[32]:

- **Repetition codes**. Let R_N denote the repetition code that repeats each bit in the message N times[33]. We model the channel as "adding[34]" a sparse noise vector $\boldsymbol{n}$ to the encoded message $\boldsymbol{t}$[35], so $\boldsymbol{r} := \boldsymbol{n} + \boldsymbol{t}$. Since f is assumed small, the optimal decoding strategy looks at the received bits N at a time and takes a majority vote:

$$\hat{s}_i \leftarrow \arg\max_{s_i} \Pr\left[s_i \mid \boldsymbol{r}_{i:i+n}\right] = \arg\max_{s_i} \Pr\left[\boldsymbol{r}_{i:i+N} \mid s_i\right] \Pr\left[s_i\right] \tag{154}$$

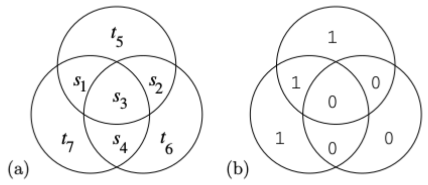
  We see that we must make assumptions about the prior probability $\Pr[s_i]$. It is common to assume all possible values of s_i (0 or 1 in this case) are equally probable. It is useful to observe the **likelihood ratio**,

$$\frac{\Pr\left[\boldsymbol{r}_{i:i+N} \mid s_i = 1\right]}{\Pr\left[\boldsymbol{r}_{i:i+N} \mid s_i = 0\right]} = \prod_{n=i}^{i+N-1} \frac{\Pr\left[r_n \mid t_n = 1\right]}{\Pr\left[r_n \mid t_n = 0\right]} = \prod_{n=i}^{i+N-1} \begin{cases} \gamma & \text{if } r_n = 1 \\ \gamma^{-1} & \text{if } r_n = 0 \end{cases} \tag{155}$$

  where we've defined $\gamma := (1-f)/f$, with f being the probability of a bit getting flipped by the channel. We want to assign $\hat{s}_i$ to the most likely **hypothesis** out of the possible s_i. If the likelihood ratio [for the two hypotheses] is greater than 1, we choose $\hat{s}_i = 1$, else we choose $\hat{s}_i = 0$.
- **Block Codes - the (7, 4) Hamming Code**. Although, by increasing the number of repetitions per bit N for our R_N repetition code can decrease the error-per-bit probability p_b, we incur a substantial decrease in the *rate of information transfer* – a factor of 1/N.

[32]A binary symmetric channel transmits each bit correctly with probability (1 - f) and incorrectly with probability f, **where f is assumed to be small.**
[33]So R_2 would encode 101 as 110011.
[34]We add in modulo 2, which is NOT the same as binary arithmetic (no carry). Addition modulo 2 is the same as doing XOR.
[35]$\boldsymbol{t}$ is the message after applying our R_N repetition code.

The **(7, 4) Hamming Code** tries to improve this by encoding *blocks* of bits at a time (instead of per-bit). It is a *linear block code* – it encodes each 4-bit block into a 7-bit block, where the additional 3 bits are linear functions of the original $K = 4$ bits. The $(7, 4)$ Hamming code has each of the extra 3 bits act as parity checks. It's easiest to show this with the illustration below:



which encodes `1000` as `1000101`.

2.1.1 Exercises

Exercise 1.2 - Error Probability for Repetition Code

Show that the error probability is reduced by the use of R_3.

[correct] The original probability of a decoding error was f (by definition). The probability of error with R_3 is

$$\begin{aligned}
\Pr[\hat{s}_i \neq s_i] &= 3f^2 \cdot (1-f) + 3f^3 && (156)\\
&\approx 3f^2(1-f) && (157)\\
&= 3f^2 - 3f^3 && (158)\\
&\approx 3f^2 && (159)\\
&< f && (160)
\end{aligned}$$

which shows that the error probability is lower than it would've been otherwise (had we not used a repetition code).

Probability, Entropy, and Inference (Ch. 2)

Written by Brandon McKinzie

[Note: Skipping most of this chapter since it's mostly introductory material.]

Notation. Some of the notation this author seems to use a lot.

- **Ensemble** X: a triple $(x, \mathcal{A}_X, \mathcal{P}_X)$, where the outcome x is the value of a R.V. which can take on one of a set of possible values ("alphabet"), $\mathcal{A}_X = \{a_1, \ldots, a_i, \ldots, a_I\}$, having probabilities $\mathcal{P}_X = \{p_1, \ldots, p_i, \ldots p_I\}$. Note that this doesn't appear technically consistent with how the author actually *uses* the term ensemble – in practice, he actually means "X is the set of all possible triples $(x, \mathcal{A}_X, \mathcal{P}_X), \forall x \in \mathcal{A}_X$, or something like that. He uses it to casually refer to the space of possibilities.

Forward Probabilities and Inverse Probabilities. Both of these involve a **generative model** of the data. In a *forward* probability problem, we want find the PDF, expectation, or some other function of a quantity that depends on the data/is *produced* by the generative process. For example, we can model a series of N coin flips as "producing" the quantity n_H, denoting the number of heads. In an *inverse* probability problem, we want to compute conditional probabilities on one or more of the *unobserved variables* in the process, *given* the observed variables.

The Likelihood Principle. For a generative model on data d given parameters $\boldsymbol{\theta}$, $\Pr[d \mid \boldsymbol{\theta}]$, and having observed a particular outcome d_1, all inferences and predictions should depend only on the function $\Pr[d_1 \mid \boldsymbol{\theta}]$.

Entropy and Related Quantities.

- **Shannon information content** of an outcome x:

$$h(x) \triangleq \lg\left(\frac{1}{P(x)}\right) \tag{161}$$

They mention the example of the unigram probabilities for each character in a document. For example, $p(z) = 0.007$, which has information content of 10.4 bits[36].

[36]Intuition digression: Recall from CS how to compute the number of bits required to represent N unique values (answer: $\lg(N)$.). Similarly, a probability of e.g 1/8 can be interpreted as "one of 8 possible outcomes", meaning that lg(1/(1/8))=3 bits are needed to encode all possible outcomes. Similarly, one could interpret $p(z)$=0.007 as "belonging to 7 of 1000 possible results". I guess in some strange world you can then say that there are $1000/7 \approx 142.86$ evenly-proportioned events like this (how do you even word this) and it would take

- **Entropy of Ensemble** X. Defined to be the average Shannon information content of an outcome.

$0 \times \lg\left(\frac{1}{0}\right) \triangleq 0$

$$H(X) \triangleq \sum_{x \in \mathcal{A}_X} \Pr[x] \lg\left(\frac{1}{\Pr[x]}\right) \tag{162}$$

$$H(X) \leq \lg(|\mathcal{A}_X|) \quad \text{with equality iff} \quad p_i = \frac{1}{\mathcal{A}_X}\ \forall i \tag{163}$$

- **Decomposability of the Entropy**. For any probability distribution $\boldsymbol{p} = \{p_1, p_2, \ldots, p_I\}$ and m (where $1 \leq m \leq I$):

$$H(\boldsymbol{p}) = H\left(\Sigma_{1:m}, \Sigma_{m+1:I}\right) \tag{164}$$

$$+ \Sigma_{1:m} H\left(\frac{p_1}{\Sigma_{1:m}}, \ldots, \frac{p_m}{\Sigma_{1:m}}\right) \tag{165}$$

$$+ \Sigma_{m+1:I} H\left(\frac{p_{m+1}}{\Sigma_{m+1:I}}, \ldots, \frac{p_I}{\Sigma_{m+1:I}}\right) \tag{166}$$

where I've let $\Sigma_{1:m} := \sum_{i=1}^{m} p_i$.

- **Kullback-Leibler Divergence** between two probability distributions $P(X)$ and $Q(X)$ that are defined over the same alphabet $\mathcal{A}_X$:

$$D_{KL}(P||Q) = \sum_x P(x) \lg \frac{P(x)}{Q(x)} \tag{167}$$

$$D_{KL}(P||Q) \geq 0 \qquad \textbf{[Gibb's Inequality]} \tag{168}$$

where, in the words of the author, "**Gibb's inequality is probably the most important inequality in this book**".

- **Convex functions and Jensen's Inequality**. A function $f(x)$ is convex over the interval $[x = a, x = b]$ if every chord of the function lies above the function. That is, $\forall x_1, x_2 \in [a, b]$ and $0 \leq \lambda \leq 1$:

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2) \tag{169}$$

$$f(\mathbb{E}[x]) \leq \mathbb{E}[f(x)] \qquad \textbf{[Jensen's Inequality]} \tag{170}$$

and we say f is **strictly convex** if, $\forall x_1, x_2 \in [a, b]$, we get equality for only $\lambda = 0$ and $\lambda = 1$.

---

log(142.86)=10.4 bits to encode all of them. Low-probability events (such as a character being z) have high information content.

Perhaps a better way to think of this is explained on the wikipedia page:

> When the content of a message is known a priori with certainty, with probability of 1, there is no actual information conveyed in the message. Only when the advance knowledge of the content of the message by the receiver is less than 10% certain does the message actually convey information. Accordingly, the amount of self-information contained in a message conveying content informing an occurrence of event, ω_n, depends only on the probability of that event.

Because this is too short to have as its own chapter...

A first inference problem. Author explores a particle decay problem of finding $\Pr[\lambda \mid \{x\}]$ where λ is the characteristic decay length and $\{x\}$ is our collection of observed decay distances. Plotting the likelihood $\Pr[\{x\} \mid \lambda]$ *as a function of* λ for any given $x \in \{x\}$ shows each has a *peak* value. The kicker is the interpretation: if each measurement $x \in \{x\}$ is independent, the total likelihood is the product of all the individual likelihoods, which can be interpreted as updating/narrowing the interval $[\lambda_a, \lambda_b]$ within which $\Pr[\{x\} \mid \lambda]$ (as a function of λ) peaks. In the words of the author's mentor:

> *what you know about λ after the data arrive is what you knew before ($\Pr[\lambda]$) and what the data told you ($\Pr[\{x\} \mid \lambda]$)*

We update our beliefs regarding the distribution of λ as we collect data.

Lessons learned from problems.

(3.8) The classic Monty Hall problem. **Be careful defining probabilities after collecting data**. My blunder: when using Bayes'theorem to get the probability of the prize being behind door $i \in \{1, 2\}$ after the host opens door 3, I failed to take into account that we chose door 1 while I was computing the evidence (the marginal probability that the host opened door 3)[37]. Quite embarrassing.

(3.9) Monty Hall problem, but an earthquake opens door 3. Although I correctly answered that, in this case, both hypotheses (the prize being behind door 1 or door 2) are equiprobable, I still failed to account for the subtle fact that the earthquake could've opened *multiple* doors. The lesson here is **always write down the probability of everything**, which just so happens to be suggested by the solution for this problem, too.

So, why did I still get the answer correct? The reason is because enumeration of probabilities wasn't necessary at all, you just needed to realize that the likelihood for the two remaining hypotheses ($\mathcal{H}_1$ and $\mathcal{H}_2$) were the *same* – the probability of observing the earthquake open door 3 and the prize not being revealed was the same for the case of the prize being behind door 1 or door 2. So maybe the real lesson here is **determine whether calculations are even needed in order to solve the given problem**, which luckily I've had drilled in my head for years from studying physics.

(3.15) Another biased coin variant. One of the best examples I've seen for favoring Bayesian methods over frequentist methods. Also, made use of the beta function:

$$\int_0^1 p^x (1-p)^y \mathrm{d}p = \frac{\Gamma(x+1)\Gamma(y+1)}{\Gamma(x+y+2)} = B(x+1, y+1) \tag{171}$$

[37]Note that, while important to recognize and understand, I could've avoided this pitfall entirely by just ignoring the evidence during calculations and normalizing after, since the evidence can be determined solely by the normalization constraint.

where B is the **beta function**, which is defined by the LHS.

Information Theory, Inference, and Learning Algorithms **November 23, 2017**

The Source Coding Theorem (Ch. 4)

Table of Contents Local *Written by Brandon McKinzie*

Overview. We will be examining the two following assertions:

1. The Shannon information content is a sensible measure of the information content of a given outcome $x = a_i$:

$$h(x = a_i) \triangleq \lg \frac{1}{p_i} \tag{172}$$

2. The entropy of an ensemble X is a sensible measure of the ensemble's average information content.

$$H(X) = \sum_i p_i \lg \frac{1}{p_i} \tag{173}$$

The weighing problem. We are instructed to ponder the following problem.

You're given 12 balls, all equal in weight except for one that is either heavier or lighter. You're also given a two-pan balance. Your task is to determine which ball is the odd ball, and in as few uses of the balance as possible. Note: each use of the balance must place the same number of balls on either side.

An interesting observation is to consider the number of possible outcomes of the weighing process. Each outcome can be one of three possibilities: equal weight, left heavier, or left lighter. After N such weighings, the number of unique possible weighing result sequences is 3^N. Note that there are $12 \times 2 = 24$ unique final answers for our task (identifying which is the odd ball, and whether it is heavier or lighter). Therefore, since we are seeking a procedure to identify which of the 24 options is the correct option with 100% accuracy, we require our weighing procedure to take on *at least* 24 unique possible results. Since $N = 3$ weighings corresponds to $3^N = 27$ possible outcomes, $N = 3$ is a *lower bound* on the number of weighings our approach will involve. It is impossible to guarantee a correct answer for $N < 3$ weighings[38].

Things I didn't consider until reading the solution:

- It's actually *not* optimal, upon observing both sides equal, to subsequently use only the balls not involved in that measurement. My initial reaction to this was "why? we already know the oddball is not any of the balls just measured, since the outcome was equal."

[38]Finally, it should be clear that, regardless of our approach, the final weighing will involve 2 balls, since we have to identify which is the oddball AND whether it is heavier/lighter AND the number of balls on the left of the scale must be the same as the right of the scale for every weighing.

The response to this reaction is: "yes, *exactly*, and we must use that information to be able to discern in the future whether, e.g., a measurement of "left side heavier" means the oddball is on the left and heavy, or if it's on the right and light – *it's useful to know that a given side of the scale does not contain the oddball before a measurement.*

- More generally, it's also not optimal to greedily search for solutions that eliminate the highest number of possibilities *in any given single step.* Another way of thinking about this is that it's undesirable for the ith measurement outcome to cause any of the 3 possible measurement outcomes to be impossible at the next stage.
- I focused a disproportionate amount of thought on handling the equal-weight measurement outcome, for whatever reason. I probably would've arrived at the solution faster if I'd actually thought about how my strategies would've handled some outcome being "left heavier" and *then considered what that strategy would put on the scale at the next step*, where the italics denote what would've illuminated the fatal flaw in all my approaches.

Guessing Games. What's the smallest number of yes/no questions needed to identify an integer x between 0 and 63? Although it was obvious to me that the solution is to successively halve the possible values of x, I found it interesting that **you can write down the list of questions independent of the answers at each step** using a basic application of modular arithmetic. In other words, you can specify the full decision tree of N nodes with just $\lg N$ questions. Nice. Also, recognize that the Shannon information content for any single outcome is $\lg \frac{1}{0.5} = 1$ bit, and thus the total Shannon information content (for our predefined 6 questions) is 6 bits, which is not-coincidentally the number of possible values that x could be before we ask any questions.

In general, if an outcome x has Shannon information content $h(x)$ number of bits, I like to interpret that as "learning the result x eliminates $2^{h(x)}$ possibilities for the final result." The battleship example follows this interpretation well. Stated another way (in the author's words):

> *The Shannon information content can be intimately connected to the size of a file that encodes the outcomes of a random experiment.*

Data Compression. A **lossy compressor** compresses some files, but maps some files to the *same* encoding. We introduce a parameter δ that describes the risk (aggressiveness of our compression) we are taking with a given compression method: δ is the probability that there will be no name for an outcome[39] x.

The smallest δ-sufficient subset

If S_δ is the smallest subset of $\mathcal{A}_X$ satisfying

$$\Pr\left[x \in S_\delta\right] \geq 1 - \delta \tag{174}$$

then S_δ is the smallest δ-sufficient subset. It can be constructed by ranking the elements of $\mathcal{A}_X$ in order of decreasing probability and adding successive elements starting from the most probable elements until the total probability is $\geq (1 - \delta)$.

- **Raw bit content** of X: $H_0(X) \triangleq \lg |\mathcal{A}_X|$. A lower bound for the number of binary questions that are always *guaranteed* to identify an outcome from the ensemble X – it simply maps each outcome to a constant-length binary string.
- **Essential bit content** of X: $H_\delta(X) \triangleq \lg |S_\delta|$. A compression code can be made by assigning a binary string of H_δ bits to each element of the smallest sufficient subset.

Finally, we can now state **Shannon's source coding theorem**: Let X be an ensemble for the random variable x with entropy $H(X) = H$ bits, and let X^N denote a sequence of identically distributed (but not necessarily independent[40]) of random variables/ensembles, $(X_1, X_2, \ldots, X_N)$.

$$(\exists N_0 \in \mathbb{Z}^+)(\forall N > N_0): \qquad \left| \frac{1}{N} H_\delta(X^N) - H \right| < \epsilon \qquad (0 < \delta < 1)\ (\epsilon > 0) \tag{175}$$

which, in English, can be read: *N i.i.d. random variables each with entropy $H(X)$ can be compressed into more than $NH(X)$ bits with negligible risk of information loss, as $N \to \infty$; conversely if they're compressed into fewer than $NH(X)$ bits it is virtually certain that information will be lost.*

[39] More specifically, if there is some subset, $\{a\}$, of unique values that x can take on but our compression method discards/ignores, then we say $\delta = \sum_i p(x = a_i)$.

[40] Actually, before the actual theorem statement, the author mentions we are now concerned with "string of N i.i.d. random variables from a single ensemble X." It's probably fair to assume this is true for the quantities in the theorem, but I'm leaving this note here as a reminder.

Typicality. The reason that large N in equation 175 corresponds to larger potential for better compression is that the subset of likely results for a string of outcomes becomes more and more concentrated relative to the number of possible sequences as N increases[41]. I just realized this is for the same fundamental reasons that entropy exists in thermodynamics – *there are just more ways to exist in a high entropy state than otherwise.* The author showed $\binom{N}{r}$ as a function of r (the number of 1s in the N-bit string). For large N, this becomes almost comically concentrated near the center (like a delta function at $N/2$) – see footnote for more details[42].

This motivates the notion of **typicality** for [a string of length N from] an arbitrary ensemble X with alphabet $\mathcal{A}_X$. For large N, we expect to find $p_i N$ occurrences of the outcome $x = a_i$. Hence the probability of such a string, and its information content, is roughly[43]

$$\Pr[\boldsymbol{x}]_{typ} = \Pr[x_1]\Pr[x_2]\cdots\Pr[x_N] \simeq p_1^{(p_1 N)} p_2^{(p_2 N)} \cdots p_I^{(p_I N)} \tag{176}$$

$$h(\boldsymbol{x})_{typ} = \lg \frac{1}{\Pr[\boldsymbol{x}]_{typ}} \simeq N \sum_{i=1}^{I} p_i \lg \frac{1}{p_i} = NH(X) \tag{177}$$

Accordingly, we define the typical elements (strings of length N) of $\mathcal{A}_X^N$ to be those elements that have probability close to 2^{-NH}. We introduce a parameter β that defines what we mean by "close," and define the set of typical elements as the **typical set** $T_{N\beta}$:

$$T_{N\beta} \triangleq \left\{ \boldsymbol{x} \in \mathcal{A}_X^N : \left| \frac{1}{N} \lg \frac{1}{P(\boldsymbol{x})} - H \right| < \beta \right\} \tag{178}$$

It turns out that whatever value of β we choose, the $T_{N\beta}$ contains almost all the probability as N increases.

[41]The author gave an example for a sequence of bits with probability of any given bit being 1 as 0.1. He showed how, although the *average* number of 1s in a sequence of N bits grew as $\mathcal{O}(N)$, the standard deviation of that average only grew as $\sqrt{N}$.

[42]The probability of getting a string with r 1s follows a binomial distribution with mean Np_1 and standard deviation $\sqrt{Np_1(1-p_1)}$. This results in an increasingly narrower distribution $P(r)$ for larger N.

[43]We appear to be assuming that each outcome x in the string $\boldsymbol{x}$ are i.i.d. (CONFIRMED)

Proving the Source Coding Theorem.

- **Setup**. We will make use of the following:
  - **Chebyshev's Inequalities**:

$$\Pr[x \geq \alpha] \leq \frac{\mathbb{E}[x]}{\alpha} \qquad \text{and} \qquad \Pr\left[(x - \mathbb{E}[x])^2 \geq \alpha\right] \leq \frac{\mathrm{Var}[x]}{\alpha} \tag{179}$$

    where α is a positive real number, and x is assumed non-negative in the first inequality[44].
  - **Weak Law of Large Numbers** (WLLN): Consider a sample $h_1, \ldots, h_N$ of N independent RVs all with common mean $\bar{h}$ and common variance σ_h^2. Let $x = \frac{1}{N}\sum_{n=1}^{N} h_n$ be their average. Then

$$\Pr\left[(x - \bar{h})^2 \geq \alpha\right] \leq \frac{\sigma_h^2}{\alpha N} \tag{180}$$

    which can be easily derived from Chebyshev's inequalities.
- **Proving 'asymptotic equipartition' principle**, i.e. that an outcome $\boldsymbol{x}$ is almost certain to belong to the typical set, approaching probability 1 for large enough N. It is a simple application of the WLLN to the random variable

$$\frac{1}{N}\lg\frac{1}{\Pr[\boldsymbol{x}]} = \frac{1}{N}\sum_{n=1}^{N}\lg\frac{1}{x_n} = \frac{1}{N}\sum_{n=1}^{N} h(x_n) \tag{181}$$

  where $\mathbb{E}[h(x_n)] = H(X)$ for all terms in the summation. Observe, then, that the definition of the typical set given in equation 178 (squaring both sides) has the same form as the definition for the WLLN. Plugging in and rearranging yields

$$\Pr[\boldsymbol{x} \in T_{N\beta}] \geq 1 - \frac{\sigma^2}{\beta^2 N} \tag{182}$$

  where $\sigma^2 \equiv \mathrm{Var}\left[\lg\frac{1}{P(x_n)}\right]$. This proves the asymptotic equipartition principle. It will also be useful to recognize that for any $\boldsymbol{x}$ in the typical set, we can rearrange equation 178 to obtain

$$2^{-N(H+\beta)} < \Pr[\boldsymbol{x}] < 2^{-N(H-\beta)} \tag{183}$$

- **Proof of SCT Part I**. Want to show that $\frac{1}{N}H_\delta(X^N) < H + \epsilon$. **TODO**
- **Proof of SCT Part II**. Want to show that $\frac{1}{N}H_\delta(X^N) > H - \epsilon$. **TODO**

[44]Notice how the two inequalities are technically the same.

Questions & Answers. Collection of my questions as I read through the chapter, which I answered upon completing it.

- **Q**: Why isn't the essential bit content of a string of N i.i.d. variables NH when $\delta = 0$?
  - **A**: I'm not entirely sure how to answer this still, but it seems the question is confused. First off, the essential bit content approaches the raw bit content as δ decreases to 0: $H_\delta \to H_0$ as $\delta \to 0$. It's important to notice that both H_δ and H_0 define an entropy where all members (S_δ^N for H_δ; $\mathcal{A}_X^N$ for H_0) are *equiprobable.* I remember asking this question wondering "what is the significance of $H_\delta(X^N)$ approaching $NH(X)$ (not a typo!) for tiny δ". The answer: for larger N, more of the probability mass is concentrated in a relatively smaller region, *with elements of that region being roughly equiprobable.* The last part is what I didn't initially realize – that **allowing for tiny δ combined with large N essentially makes it so that $S_\delta^N \approx T_{N\beta}$.**
- **Q**: Why aren't the most probable elements necessarily in the typical set?
  - **A**: In the limit of $N \to \infty$, *they are*, since in that limit, all elements are in the typical set and they're equiprobable. However, in essentially any real case, we can imagine that some elements will be too unlikely to be found within the typical set, which necessarily requires that there exist elements with probability too high to be in the typical set. Remember that the typical set is basically *defined* such that all elements have probability within the range given in equation 183.

Monte Carlo Methods (Ch. 29)

Overview. The aims of Monte Carlo methods are to solve one or both of the following:

1. Generate samples $\{\boldsymbol{x}^{(r)}\}_{r=1}^{R}$ from a given probability distribution $\Pr[\boldsymbol{x}]$.
2. Estimate expectations of function under $\Pr[\boldsymbol{x}]$, for example

$$\boldsymbol{\Phi} = \mathbb{E}_{\boldsymbol{x}\sim\Pr[\boldsymbol{x}]}[\phi(\boldsymbol{x})] \equiv \int d^N\boldsymbol{x} \Pr[\boldsymbol{x}]\,\phi(\boldsymbol{x}) \tag{184}$$

where it's assumed that $\Pr[\boldsymbol{x}]$ is sufficiently complex that we can't evaluate such expectations by exact methods.

Note that we can concentrate on the first problem (sampling), since we can use it to solve the second problem (estimating an expectation) by using the random samples $\{\boldsymbol{x}^{(r)}\}_{r=1}^{R}$ to give the estimator

$$\hat{\boldsymbol{\Phi}} \equiv \frac{1}{R}\sum_r \phi(\boldsymbol{x}^{(r)}) \tag{185}$$

where

$$\mathbb{E}\left[\hat{\boldsymbol{\Phi}}\right] = \frac{1}{R}\sum_r \mathbb{E}_{\boldsymbol{x}^{(r)}\sim\Pr[\boldsymbol{x}]}\left[\phi(\boldsymbol{x}^{(r)})\right] = \boldsymbol{\Phi} \tag{186}$$

$$\lim_{R\to\infty} \mathrm{Var}\left[\hat{\boldsymbol{\Phi}}\right] = \lim_{R\to\infty} \frac{\sum_r \left(\phi(\boldsymbol{x}^{(r)}) - \boldsymbol{\Phi}\right)^2}{R-1} = \frac{\sigma^2}{R} \equiv \frac{1}{R}\int d^N\boldsymbol{x} \Pr[\boldsymbol{x}]\,(\phi(\boldsymbol{x}) - \boldsymbol{\Phi})^2 \tag{187}$$

Importance Sampling. A generalization of [naively] uniformly sampling $\boldsymbol{x}$ in order to approximate equation 184. We assume henceforth that we are able to evaluate (for now, a 1-D) $\Pr[x]$ at any point x at least within a multiplicative constant; thus we can evaluate a function $P^*(x)$ such that $P(x) = P^*(x)/Z$. We assume we have some simpler Q(x), called the **sampler density**, from which we *can* generate samples from and evaluate up to a multiplicative constant, $Q(x) = Q^*(x)/Z_Q$. We construct an approximation for our estimator in equation 185 via sampling from $Q(x)$ and computing:

$$\hat{\boldsymbol{\Phi}} = \frac{\sum_r w_r \phi(x^{(r)})}{\sum_r w_r} \qquad \text{where} \qquad w_r \equiv \frac{P^*(x^{(r)})}{Q^*(x^{(r)})} \tag{188}$$

Rejection Sampling. In addition to the assumptions in importance sampling, we now also assume that we know the value of a constanct c such that

$$\forall x: \quad cQ^*(x) > P^*(x) \tag{189}$$

1. Generate sample x from proposal density $Q(x)$, and evaluate $cQ^*(x)$.
2. Generate a uniformly distributed random variable u from the interval $[0, cQ^*(x)]$.
3. If $u > P^*(x)$, then reject x; else, accept x and add it to our set of samples $\{x^{(r)}\}$.

Metropolis-Hastings Method. Proposal density Q now depends on the current state $x^{(t)}$.

1. Sample tentative next state x' from proposal density $Q(x'; x^{(t)})$.
2. Compute:

$$a = \frac{P^*(x')}{P^*(x^{(t)})} \frac{Q(x^{(t)}; x')}{Q(x'; x^{(t)})} \tag{190}$$

3. If $a \geq 1$, the new state is accepted. Otherwise, the new state is accepted with probability a.
4. If accepted, we set $x^{(t+1)} = x'$. If rejected, we set $x^{(t+1)} = x^{(t)}$.

Information Theory, Inference, and Learning Algorithms **November 15, 2017**

Variational Methods (Ch. 33)

Table of Contents Local *Written by Brandon McKinzie*

Probability Distributions in Statistical Physics[45]. Consider the common situation below, where the state vector $\boldsymbol{x} \in \{-1, +1\}^N$:

$$\Pr[\boldsymbol{x} \mid \beta, \boldsymbol{J}] = \frac{e^{-\beta E(\boldsymbol{x};\boldsymbol{J})}}{Z(\beta, \boldsymbol{J})} \tag{191}$$

$$E(\boldsymbol{x};\boldsymbol{J}) = -\frac{1}{2}\sum_{m,n} J_{mn}x_m x_n - \sum_n h_n x_n \tag{192}$$

$$Z(\beta, \boldsymbol{J}) = \sum_{\boldsymbol{x}} e^{-\beta E(\boldsymbol{x};\boldsymbol{J})} \tag{193}$$

Note that evaluating $E(\boldsymbol{x};\boldsymbol{J})$ for a given $\boldsymbol{x}$ takes polynomial time in the number of spins N, and evaluating Z is $\mathcal{O}(2^N)$. Variational free energy minimization is a method for approximating the complex distribution $P(\boldsymbol{x})$ by a simpler ensemble $Q(\boldsymbol{x};\boldsymbol{\theta})$ parameterized by adjustable $\boldsymbol{\theta}$.

Variational Free Energy. How do we find/evaluate Q? The objective function chosen to measure the quality of the approximation is the **variational free energy**, $\widetilde{F}(\boldsymbol{\theta})$:

$$\beta\widetilde{F}(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} Q(\boldsymbol{x};\boldsymbol{\theta}) \ln \frac{Q(\boldsymbol{x};\boldsymbol{\theta})}{e^{-\beta E(\boldsymbol{x};\boldsymbol{J})}} \tag{194}$$

$$= \beta\mathbb{E}_{\boldsymbol{x}\sim Q}\left[E(\boldsymbol{x};\boldsymbol{J})\right] - H(Q) \tag{195}$$

$$= D_{KL}\left(Q||P\right) + \beta F \tag{196}$$

where $F \triangleq -\ln Z(\beta, J)$ is the true free energy. It's not immediately clear why this approximation Q is more tractable than P – for that we turn to an example below.

[45] Yay!

Variational Free Energy Minimization for Spin Systems. For the system with energy function given in equation 192, consider the *separable* approximating distribution,

$$Q(\boldsymbol{x};\boldsymbol{a}) = \frac{e^{\sum_n a_n x_n}}{Z_Q} = \frac{\prod_n e^{a_n x_n}}{\sum_{\boldsymbol{x}'} \prod_{n'} e^{a_n x'_{n'}}} = \frac{\prod_n e^{a_n x_n}}{\sum_{x'_1} e^{a_1 x'_1} \sum_{x'_2} e^{a_2 x'_2} \cdots \sum_{x'_n} e^{a_n x'_n}} \tag{197}$$

To compute $\widetilde{F}$, we must compute the mean energy and entropy under Q.

→ **Mean Energy**. Given our definition of Q, and the fact that each $x_n = \pm 1$, the mean value of any x_n is $\bar{x} = \tanh(a_n) = 2q_n - 1$, where $q_n \equiv Q(x_n = 1)$.

$$\mathbb{E}_{\boldsymbol{x}\sim Q}\left[E(\boldsymbol{x};\boldsymbol{J})\right] = \sum_{\boldsymbol{x}} Q(\boldsymbol{x};\boldsymbol{a}) \left[-\frac{1}{2}\sum_{m,n} J_{mn} x_m x_n - \sum_n h_n x_n\right] \tag{198}$$

$$= -\frac{1}{2}\sum_{m,n} J_{mn} \bar{x}_m \bar{x}_n - \sum_n h_n \bar{x}_n \tag{199}$$

→ **Entropy**. Recall that if $Q(\boldsymbol{x};\boldsymbol{a}) = \prod_n Q(x_n; a_n)$ (i.e. Q is separable), that $H(\boldsymbol{x}) = \sum_n H(x_n)$, so we have

$$H_Q(\boldsymbol{x}) = \sum_{\boldsymbol{x}} Q(\boldsymbol{x};\boldsymbol{a}) \ln \frac{1}{Q(\boldsymbol{x};\boldsymbol{a})} \tag{200}$$

$$= \sum_n Q(x_n = 1; a_n) \ln \frac{1}{Q(x_n = 1; a_n)} + Q(x_n = 0; a_n) \ln \frac{1}{Q(x_n = 0; a_n)} \tag{201}$$

$$= \sum_n q_n \ln \frac{1}{q_n} + (1 - q_n) \ln \frac{1}{1 - q_n} \tag{202}$$

So the variational free energy is given by

$$\beta \widetilde{F}(\boldsymbol{a}) = \beta \mathbb{E}_{\boldsymbol{x}\sim Q}\left[E(\boldsymbol{x};\boldsymbol{J})\right] - H_Q(\boldsymbol{x}) \tag{203}$$

$$= \beta\left(-\frac{1}{2}\sum_{m,n} J_{mn}\bar{x}_m\bar{x}_n - \sum_n h_n \bar{x}_n\right) - \left(\sum_n q_n \ln\frac{1}{q_n} + (1-q_n)\ln\frac{1}{1-q_n}\right) \tag{204}$$

Remember, our goal is to find parameters $\boldsymbol{a}$ that minimize $\widetilde{F}(a)$:

$$\beta \frac{\partial \widetilde{F}}{\partial a_m} = 2\left(\frac{\partial q_m}{\partial a_m}\right)\left[-\beta\left(\sum_n J_{mn}\bar{x}_n + h_m\right) + a_m\right] \tag{205}$$

which, when set to zero, yields

$$a_m \leftarrow \beta\left(\sum_n J_{mn}\bar{x}_n + h_m\right) \tag{206}$$

$$\bar{x}_n = \tanh(a_n) \tag{207}$$

Machine Learning: A Probabilistic Perspective

Contents

Machine Learning: A Probabilistic Perspective **August 07, 2018**

Probability (Ch. 2)

Table of Contents Local *Written by Brandon McKinzie*

Kevin P. Murphy (2012). Probability.
Machine Learning: A Probabilistic Perspective.

Continuous Random Variables (2.2.5). Let X be a continuous RV. We usually want to know the probability that X lies in the interval $a \le X \le b$, which is given by

$$p(a < X \le b) = p(X \le b) - p(X \le a) \tag{208}$$

Define the **cumulative distribution function** (cdf) $F(q) \triangleq p(X \le q)$, and the **probability density function** (pdf) $f(x) \triangleq \frac{d}{dx}F(x)$.

Transformation of Random Variables (2.6). In what follows, we have some $x \sim p_x$ and $y = f(x)$. How should we think about $p_y(y)$? For discrete RV x, we just sum up the probability mass for all x such that $f(x) = y$,

$$p_y(y) = \sum_{x: f(x)=y} p_x(x) \tag{209}$$

If $\mathbf{x}$ is *continuous*, we must instead work with the cdf,

$$P_y(y) \triangleq P(Y \le y) = P(f(X) \le y) = P(X \in \{x \mid f(x) \le y\}) \tag{210}$$

If f is monotonic (and hence invertible), then we can derive the pdf $p_y(y)$ from the pdf $p_x(x)$ by taking derivatives as follows:

$$p_y(y) \triangleq \frac{d}{dy}P_y(y) = \frac{d}{dy}P_x(f^{-1}(y)) = \frac{dx}{dy}p_x(x) \tag{211}$$

and, since we only work with integrals over densities (i.e. overall sign does not matter), it is convention to take the absolute value of $\frac{dx}{dy}$ in the above formula. In the multivariate case, the Jacobian matrix $[\boldsymbol{J}_{\boldsymbol{y}\to\boldsymbol{x}}]_{i,j} \triangleq \frac{\partial x_i}{\partial y_j}$ is used:

$$p_y(\boldsymbol{y}) = p_x(\boldsymbol{x}) \Big| \det \boldsymbol{J}_{\boldsymbol{y}\to\boldsymbol{x}} \Big| \tag{212}$$

Central Limit Theorem (2.6.3). Consider N i.i.d. RVs each with arbitrary pdf $p(x_i)$, mean μ, and covariance σ^2. Let $S_N = \sum_i X_i$ denote the sum over the N RVs. The **central limit theorem** states that

$$\lim_{N\to\infty} S_N \sim \mathcal{N}(N\mu, N\sigma^2) \tag{213}$$

$$\text{or equivalently} \qquad \lim_{N\to\infty} \sqrt{N}(\bar{X} - \mu) \sim \mathcal{N}(0, \sigma^2) \tag{214}$$

3.1.1 Exercises

Exercise 2.1

(a) [correct] P(oneIsGirl | hasAtLeastOneBoy) = 2/ 3. Use muh intuition.

(b) [correct] P(otherIsGirl | weSawOneIsABoy) = P(childIsGirl) = 1 / 2. The other being a girl/boy is independent of the fact that the other is a boy. All about how it is phrased, yo.

Exercise 2.2 - Variance of a sum

Show that $\mathrm{Var}[X+Y] = \mathrm{Var}[X] + \mathrm{Var}[Y] + 2\mathrm{Cov}[X,\ Y]$.
[correct] Math approach:

$$\begin{aligned} \mathrm{Var}[X+Y] &\triangleq \mathbb{E}\left[((X+Y) - \mathbb{E}[X] - \mathbb{E}[Y])^2\right] && (215) \\ &= \mathbb{E}\left[((X - \mathbb{E}[X]) + (Y - \mathbb{E}[Y]))^2\right] && (216) \\ &= \mathrm{Var}[X] + \mathrm{Var}[Y] + 2\mathrm{Cov}[X,\ Y] && (217) \end{aligned}$$

Intuition approach: If X and Y are uncorrelated, it is intuitive that their sum should have variance equal to the sum of the individual variances. If there is some linear dependence between the two, we'd expect it to either increase (if positively correlated) or decrease (if negatively correlated) the variance of their sum.

Machine Learning: A Probabilistic Perspective **August 07, 2018**

Generative Models for Discrete Data (Ch. 3)

Table of Contents Local *Written by Brandon McKinzie*

Kevin P. Murphy (2012). Generative Models for Discrete Data.
Machine Learning: A Probabilistic Perspective.

Bayesian Concept Learning (3.1). The interesting notion of learning a *concept* $\mathcal{C}$, such as “all prime numbers”, by only seeing positive examples $x \in \mathcal{C}$. How should we approach predicting whether a new test case $\widetilde{x}$ belongs to the concept $\mathcal{C}$? Well, what we're actually doing is as follows: Given an initial **hypothesis space** $\mathcal{H}$ of concepts, we collect data $\mathcal{D}$ that gradually narrows the down the subset of $\mathcal{H}$ consistent with our data. We also need to address how we (humans) will weigh certain hypotheses differently even if they are both entirely consistent with the evidence. The Bayesian approach can be summarized as follows (no particular order):

- **Likelihood**. Probability of observing $\mathcal{D}$ given a particular hypothesis h. In the simple but illustrative case where the data is sampled from a uniform distribution over the *extension*[46] of a concept (a.k.a. the *strong sampling assumption*). The probability here of sampling N items independently under hypothesis h is

$$p(\mathcal{D} \mid h) = \left[\frac{1}{|h|}\right]^N \tag{218}$$

  which elucidates how *the model favors the smallest hypothesis space consistent with the data*[47].
- **Prior**. Using just the likelihood can mean we fall prey to contrived/overfitting hypotheses that basically just enumerate the data. The prior $p(h)$ allows us to specify properties about how the learned hypothesis ought to look.
- **Posterior**. This is just the likelihood times the prior, normalized [to one]. In general, as we collect more and more data, the posterior tends toward a Dirac measure peaked at the MAP estimate:

$$p(h \mid \mathcal{D}) \to \delta_{\hat{h}^{MAP}}(h) \qquad \text{where} \tag{219}$$

$$\hat{h}^{MAP} = \arg\max_h p(h \mid \mathcal{D}) = \arg\max p(\mathcal{D} \mid h)p(h) = \arg\max_h \left[\log p(\mathcal{D} \mid h) + \log p(h)\right] \tag{220}$$

[46]The extension of a concept is just the set of numbers that belong to it.
[47]A result commonly known as **Occam's razor** or the **size principle**.

The Beta-Binomial Model (3.3). In the previous section we considered inferring some discrete distribution over integers; now we will turn to a continuous version. Consider the problem of inferring the probability that a coin shows up heads, given a series of observed coin tosses.

- **Likelihood**. As should be familiar, we'll model the outcome of each toss $X_i \in \{1, 0\}$ indicating heads or tails with $X_i \sim \text{Ber}(\theta)$. Assuming the tosses are i.i.d, this gives us $p(\mathcal{D} \mid \theta) \propto \theta^{N_1}(1-\theta)^{N_0}$, where N_1 and N_0 are the **sufficient statistics** of the data, since $p(\theta \mid \mathcal{D})$ can be entirely modeled as $p(\theta \mid N_1, N_0)$.
- **Prior**. We technically just need a prior $p(\theta)$ with support over the interval $[0, 1]$, but it would be convenient if the prior had the same form as the likelihood, i.e. $p(\theta) \propto \theta^{\gamma_1}(1-\theta)^{\gamma_2}$ for some hyperparameters γ_1 and γ_2 This is satisfied by the **Beta distribution**[48] This would also result in the posterior having the same form as the prior, meaning the prior is a **conjugate prior** for the corresponding likelihood.
- **Posterior**. As mentioned, the posterior corresponding to a Bernoulli/binomial likelihood with a beta prior is itself a beta distribution:

  $$p(\theta \mid \mathcal{D}) \propto \text{Bin}(N_1 \mid \theta, N_0 + N_1)\text{Beta}(\theta \mid a, b) \propto \text{Beta}(\theta \mid N_1 + a, N_0 + b) \tag{221}$$

  By either reading off from a table or deriving via calculus, we can obtain the following properties for our Beta posterior:

  $$\text{[mode]} \qquad \hat{\theta}_{MAP} = \frac{a + N_1 - 1}{a + b + N - 2} \tag{222}$$

  $$\text{[mean]} \qquad \bar{\theta} = \frac{a + N_1}{a + b + N} = \lambda m_1 + (1-\lambda)\hat{\theta}_{MLE} \tag{223}$$

  where $m_1 = \frac{a}{a+b}$ is the prior mean. The last form captures the notion that the posterior is a compromise between what we previously believed and what the data is telling us.

The Dirichlet-Multinomial Model (3.4). We now generalize further to inferring the probability that a die with K sides comes up as face k.

- **Likelihood**. As before, we assume a specific observed sequence $\mathcal{D}$ of N dice roles. Assuming the data is i.i.d., the likelihood has the form

  $$p(\mathcal{D} \mid \boldsymbol{\theta}) = \prod_{k=1}^{K} \theta_k^{N_k} \tag{224}$$

  which is the likelihood for the multinomial model up to an irrelevant constant factor.
- **Prior**. We'd like to find a conjugate prior for our likelihood, and we need it to have support over the K-dimensional *probability simplex*[49]. The Dirichlet distribution satisfies

[48] You may be wondering, why not the Bernoulli distribution? It trivially has the same form as the Bernoulli distribution, eh? Then, pause and actually think about what you're saying for five seconds. You want to model a prior on θ with a Bernoulli distribution? You do realize that the support for a Bernoulli is in $k \in \{0, 1\}$. It's the opposite domain entirely. We want something that "looks" like a Bernoulli but is a distribution over θ, NOT the value(s) of x.

[49] The K-dimensional probability simplex is the $(K-1)$th dimensional simplex determined by the unit vectors $e_1, \ldots, e_K \in \mathbb{R}^K$, i.e. the set of vectors $\boldsymbol{x}$ such that $x_i \geq 0$ and $\sum_i x_i = 1$.

both of these and is defined as

$$\mathrm{Dir}(\boldsymbol{\theta} \mid \boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{k=1}^{K} \theta_k^{\alpha_k - 1} \tag{225}$$

where $\boldsymbol{\theta} \in S_K$ is a built-in assumption.

- **Posterior**. By construction, this will also be Dirichlet. Note that to derive the MAP estimator we must enforce the constraint that $\sum_k \theta_k = 1$, which can be done with a **Lagrange multiplier**. The constrained objective (the Lagrangian) is

$$\ell(\boldsymbol{\theta}, \lambda) = \sum_k N_k \log \theta_k + \sum_k (\alpha_k - 1) \log \theta_k + \lambda \left(1 - \sum_k \theta_k\right) \tag{226}$$

  To get $\hat{\theta}_{MAP}$, we'd take derivatives w.r.t. λ and each θ_k, do some substitutions and solve.

Example: Language Models with Bag of Words

Given a sequence of words, predict the most likely next word. Assume that the ith word $X_i \in \{1, \ldots, K\}$ (where K is the size of our vocab) is sampled indep from all others using a Cat($\boldsymbol{\theta}$) (multinoulli) distribution. This is the BoW model.

- -

My attempt: We need to derive the form of posterior predictive $p(X_{i+1} \mid X_1, \ldots, X_i)$ where $\boldsymbol{\theta} \in S_K$. First, I know that the posterior is

$$p(\boldsymbol{\theta} \mid X_1, \ldots, X_i) \propto \mathrm{Dir}(\boldsymbol{\theta} \mid \boldsymbol{\alpha}) P(X_1, \ldots, X_i \mid \boldsymbol{\theta}) = \mathrm{Dir}(\boldsymbol{\theta} \mid \alpha_1 + N_1, \ldots, \alpha_K + N_K) \tag{227}$$

and so I can derive the posterior predictive in the usual way, while also exploiting the fact that all $X_i \perp X_j$,

$$\begin{aligned} p(X_{i+1} = k \mid X_1, \ldots, X_i) &= \int p(X_{i+1} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid X_1, \ldots, X_i) \mathrm{d}\boldsymbol{\theta} && (228) \\ &= \int \theta_k p(\theta_k, \boldsymbol{\theta}_{-k} \mid X_1, \ldots, X_i) \mathrm{d}\theta_k \mathrm{d}\boldsymbol{\theta}_{-k} && (229) \\ &= \int \theta_k p(\theta_k \mid X_1, \ldots, X_i) \mathrm{d}\theta_k && (230) \\ &= \mathbb{E}\left[\theta_k \mid X_1, \ldots, X_i\right] && (231) \\ &= \frac{\alpha_k + N_k}{\sum_j \alpha_j + N_j} && (232) \end{aligned}$$

which also shows another nice example of Bayesian smoothing.

Naive Bayes Classifiers (3.5). For classifying vectors of discrete-valued features, $\mathbf{x} \in \{1, \dots, K\}^D$. Assumes features are conditionally independent given the class label:

$$p(\boldsymbol{x} \mid y = c, \boldsymbol{\theta}) = \prod_{j=1}^{D} p(x_j \mid y = c, \theta_{j,c}) \tag{233}$$

with parameters $\boldsymbol{\theta} \in \mathbb{R}^{D \times |\mathcal{Y}|}$[50]. We can model the individual class-conditional densities with the multinoulli distribution. If we were modeling real-valued features, we could instead use a Gaussian distribution.

- **MLE fitting**. Our log-likelihood is

$$\log p(\mathcal{D} \mid \boldsymbol{\theta}) = \sum_c N_c \log \pi_c + \sum_i^N \sum_j^D \log p(x_j^{(i)} \mid y^{(i)}, \theta_{j,y^{(i)}}) \tag{234}$$

  where $\pi_c = p(y = c)$ are the class priors[51]. We see that we can optimize the class priors separately from the others, and that they have the same form as the multinomial likelihood we used in the last section. We already know that the MLE for these are $\hat{\pi}_c = N_c/N$ (remember this involves using a Lagrangian). Let's assume next that we're working in the case of binary features ($x_j \mid c \sim \text{Ber}(\theta_{j,c})$). This results in MLE estimates $\hat{\theta}_{j,c} = N_{j,c}/N_c$.

[50] You could also generalize this to having some number of params N associated with each pairwise (j, c). It's also important to recognize that this is the first model of this chapter where the input $\boldsymbol{x}$ is a *vector*, and thus introduces pairwise parameters.

[51] We only see the class prior parameters here because they appear in the likelihood of generative classifiers, since $p(x, y) = p(y)p(x \mid y)$. We don't see the priors for θ that aren't class priors because MLE is only concerned with maximizing the likelihood, not the posterior (which would contain those priors).

3.2.1 Exercises

MLE for uniform distribution (3.8)

The density function for the uniform distribution centered on 0 with width 2a is

$$p(x) = \frac{1}{2a}\mathbb{1}\{x \in [-a, a]\}$$

Remember this is for continuous x, and $p(x)$ is interpreted as the derivative of the corresponding CDF $P(x)$.

a. Given data $x_1, \ldots, x_n$, find $\hat{a}_{MLE}$. So there are a few ways of doing this. We can get the answer pretty quick with intuition, and not-so-quick by grinding through the math. **Quick-n-easy**: If you were paying attention to the chapter, you'd instantly remember that MLE is all about finding the smallest hypothesis space consistent with the data. It should then be obvious that $\hat{a}_{MLE} = \max |x_i|$. **Slightly more rigorous**. We can also solve a constrained optimization problem, with constraint that $a \geq \max |x_i|$, since we must require our solution to assign nonzero probability to any of our observations.

$$\hat{a}_{MLE} = \arg\max_a \log p(x_1, \ldots, x_n \mid a) + \lambda(a - \max |x_i|) \tag{235}$$

The rest is mechanical: (1) take deriv wrt λ, (2) deriv wrt a and set to zero, (3) solve for a as a function of λ, (4) solve for λ by substituting previous step's results into contstraint, yielding that $\lambda \leq n/|x_{max}|$, (5) plug result for λ into result from step 3 to obtain result that $a \geq x_{max}$. In the limit of many samples, the first term becomes more important in the optimization, which decreases as a increases, and so we choose the lowest value of a that satisfies the constraints: $a := x_{max}$.

b. What probability would the model assign to a new data point x_{n+1} using $\hat{a}$. We are obviously meant to trivially answer that it will assign the density with $a = \hat{a}$. However, I take objection to this question, since it makes no sense to evaluate a density at a single point $p(x)$.

c. Do you see any problem with this approach? Yes, it extremely overfits to the data. We'll assign zero probabilities to any points outside the interval of observations, and the same probability to anything in that interval. We should do a more Bayesian approach instead.

Machine Learning: A Probabilistic Perspective **August 11, 2018**

Gaussian Models (Ch. 4)

Table of Contents Local *Written by Brandon McKinzie*

Kevin P. Murphy (2012). Gaussian Models.
Machine Learning: A Probabilistic Perspective.

Basics (4.1). I'll be filling in the gaps that the book leaves out in its derivations, as a way of reviewing the relevant linear algebra/calculus/etc. For notation's sake, here how the author writes the MVN in D dimensions:

$$\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \triangleq \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right\} \tag{236}$$

To get a better understanding, let's inspect the eigendecomposition of $\boldsymbol{\Sigma}$. We know that $\boldsymbol{\Sigma}$ is positive definite[52], and therefore the eigendecomposition $\boldsymbol{\Sigma} = \boldsymbol{U\Lambda U}^T$ exists, where $\boldsymbol{U}$ is an orthonormal matrix of eigenvectors. By the invertible matrix theorem, we therefore know that $\boldsymbol{U}$ is invertible. Since $\boldsymbol{\Sigma}$ is p.d., it's eigenvalues are all positive, and thus $\boldsymbol{\Lambda}$ is also invertible. We can then apply the basic definition for an invertible matrix to write

Decomposition of $\boldsymbol{\Sigma}$

$$\boldsymbol{\Sigma}^{-1} = \boldsymbol{U\Lambda}^{-1}\boldsymbol{U}^T = \sum_{i=1}^{D} \frac{1}{\lambda_i}\boldsymbol{u}_i\boldsymbol{u}_i^T \tag{237}$$

Remember, an orthonormal matrix satisfies $U^T = U^{-1}$.

where $\boldsymbol{u}_i$ is the ith eigenvector and ith *column* of $\boldsymbol{U}$. We can use this to rewrite

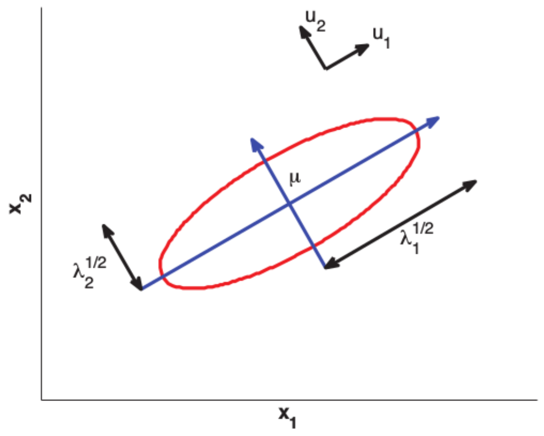
Rewriting in form of an ellipse.

$$(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) = (\boldsymbol{x}-\boldsymbol{\mu})^T\left(\sum_i^D \frac{1}{\lambda_i}\boldsymbol{u}_i\boldsymbol{u}_i^T\right)(\boldsymbol{x}-\boldsymbol{\mu}) \tag{238}$$

$$= \sum_i^D \frac{1}{\lambda_i}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{u}_i\boldsymbol{u}_i^T(\boldsymbol{x}-\boldsymbol{\mu})^T \tag{239}$$

$$= \sum_i^D \frac{1}{\lambda_i}y_i^2 \tag{240}$$

[52] We know this because all covariance matrices of any random vector X are symmetric p.s.d., and the additional requirement that Σ^{-1} exists means that it is p.d.



where $y_i \triangleq \boldsymbol{u}_i^T(\boldsymbol{x} - \boldsymbol{\mu})$. The fascinating insight is recalling that the equation for an ellipse in 2D is

$$\frac{y_1^2}{\lambda_1} + \frac{y_2^2}{\lambda_2} = 1 \tag{241}$$

Hence we see that the contours of equal probability density of a Gaussian lie along ellipses, as illustrated above. The eigenvectors determine the orientation of the ellipse, and the eigenvalues determine how elongated it is.

Maximum Entropy Derivation of the Gaussian (4.1.4). Recall that the exponential family can be derived as the family of distributions $p(\boldsymbol{x})$ that maximizes $H(p)$ subject to constraints that the moments of p match some set of empirical moments F_k specified by us. It turns out that *the MVN is the distribution with maximum entropy subject to having a specified mean and covariance.* Consider the zero-mean MVN and its entropy[53]:

$$p(\boldsymbol{x}) = \frac{1}{Z} \exp\left\{ -\frac{1}{2} \boldsymbol{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{x} \right\} \tag{242}$$

$$h(p) = \frac{1}{2} \ln \left[(2\pi e)^D \det \boldsymbol{\Sigma} \right] \tag{243}$$

Let $p = \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ above and let $q(\boldsymbol{x})$ be any density satisfying $\int q(\boldsymbol{x}) x_i x_j \mathrm{d}\boldsymbol{x} = \Sigma_{ij}$. The result is based on the fact that $h(q)$ must be less than or equal to $h(p)$. This can be shown by evaluating $D_{KL}(q||p)$ and recalling that D_{KL} is always greater than or equal to zero.

[53]For derivation, see this wonderful answer on stackexchange.

Gaussian Discriminant Analysis (4.2). With generative classifiers, it is common to define the class-conditional density as a MVN,

$$p(\boldsymbol{x} \mid y{=}c, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \tag{244}$$

which results in a technique called (Gaussian) **discriminant**[54] **analysis**. If Σ_c is diagonal, this is just a form of naive Bayes[55]. We classify some feature vector $\boldsymbol{x}$ using the decision rule:

$$\hat{y}(\boldsymbol{x}) = \arg\max_c \left[\log p(y{=}c \mid \boldsymbol{\pi}) + \log p(\boldsymbol{x} \mid y{=}c, \boldsymbol{\theta})\right] \tag{245}$$

If we have uniform prior over classes, we can classify a new test vector as follows:

$$\hat{y}(\boldsymbol{x}) = \arg\min_c (\boldsymbol{x} - \boldsymbol{\mu}_c)^T \boldsymbol{\Sigma}_c^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_c) \tag{246}$$

Linear Discriminant Analysis (LDA). The special case where all covariance matrices are the same, $\boldsymbol{\Sigma}_c = \boldsymbol{\Sigma}$. Now the quadratic term $\boldsymbol{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{x}$ is independent of c and thus is not important for classification. Instead we end up with the much simpler,

$$p(y{=}c \mid \boldsymbol{x}, \boldsymbol{\theta}) = \frac{e^{\beta_c^T \boldsymbol{x} + \gamma_c}}{\sum_{c'} e^{\beta_{c'}^T \boldsymbol{x} + \gamma_{c'}}} = \mathcal{S}(\boldsymbol{\eta})_c \tag{247}$$

$$\boldsymbol{\beta}_c := \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_c \tag{248}$$

$$\gamma_c := -\frac{1}{2} \boldsymbol{\mu}_c \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_c + \log \pi_c \tag{249}$$

where $\mathcal{S}$ is the familiar **softmax**. **TODO**: come back and compare/contrast LDA with multinomial logistic regression after reading chapter 8.

Inference in Jointly Gaussian Distributions (4.3). Suppose $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)$ is jointly Gaussian with parameters

$$\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{pmatrix} \tag{250}$$

Then, the marginals are given by

$$p(\boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{x}_1 \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11}) \tag{251}$$

$$p(\boldsymbol{x}_2) = \mathcal{N}(\boldsymbol{x}_2 \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_{22}) \tag{252}$$

and the posterior conditional is given by

[54] Don't confuse the word "discriminant" for "discriminative" – we are still in a generative model setting. See 8.6 for details on the distinction.

[55] This is easy to show. If diagonal, then $p(\boldsymbol{x} \mid y)$ factorizes. We know the ith item in the product corresponds to $p(x_i \mid c)$ by considering how simple it is to compute marginals for Gaussians with diagonal Σ.

$$
\begin{aligned}
p(\boldsymbol{x}_1 \mid \boldsymbol{x}_2) &= \mathcal{N}(\boldsymbol{x}_1 \mid \boldsymbol{\mu}_{1|2}, \boldsymbol{\Sigma}_{1|2}) \\
\text{where} \qquad \boldsymbol{\mu}_{1|2} &= \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2) \\
\boldsymbol{\Sigma}_{1|2} &= \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21} = \boldsymbol{\Lambda}_{11}^{-1}
\end{aligned} \tag{4.69}
$$

where $\boldsymbol{\Lambda} := \boldsymbol{\Sigma}^{-1}$. Notice that the conditional covariance is a *constant* matrix independent of $\boldsymbol{x}_2$. The proof here relies on Schur complements (see Appendix).

Information Form (4.3.3). Thus far, we've been working in terms of the **moment parameters** $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. We can also rewrite the MVN in terms of its **natural (canonical) parameters** $\boldsymbol{\Lambda}$ and $\boldsymbol{\xi}$,

$$
\mathcal{N}_c(\boldsymbol{x} \mid \boldsymbol{\xi}, \boldsymbol{\Lambda}) = (2\pi)^{-D/2}|\boldsymbol{\Lambda}|^{\frac{1}{2}} \exp\left\{-\frac{1}{2}\left(\boldsymbol{x}^T\boldsymbol{\Lambda}\boldsymbol{x} + \boldsymbol{\xi}^T\boldsymbol{\Lambda}^{-1}\boldsymbol{\xi} - 2\boldsymbol{x}^T\boldsymbol{\xi}\right)\right\} \tag{253}
$$

$$
\text{where} \quad \boldsymbol{\Lambda} \triangleq \boldsymbol{\Sigma}^{-1} \quad \text{and} \quad \boldsymbol{\xi} \triangleq \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \tag{254}
$$

where $\mathcal{N}_c$ is how we'll denote "in canonical form." The marginals and conditionals in canonical form are

$$
p(\boldsymbol{x}_2) = \mathcal{N}_c(\boldsymbol{x}_2 \mid \boldsymbol{\xi}_2 - \boldsymbol{\Lambda}_{21}\boldsymbol{\Lambda}_{11}^{-1}\boldsymbol{\xi}_1,\ \boldsymbol{\Lambda}_{22} - \boldsymbol{\Lambda}_{21}\boldsymbol{\Lambda}_{11}^{-1}\boldsymbol{\Lambda}_{12}) \tag{255}
$$

$$
p(\boldsymbol{x}_1 \mid \boldsymbol{x}_2) = \mathcal{N}_c(\boldsymbol{x}_1 \mid \boldsymbol{\xi}_1 - \boldsymbol{\Lambda}_{12}\boldsymbol{x}_2,\ \boldsymbol{\Lambda}_{11}) \tag{256}
$$

and we see that **marginalization is easy in moment form, while conditioning is easier in information form**.

Linear Gaussian Systems (4.4). Suppose we have a hidden variable $\boldsymbol{x}$ and a noisy observation of it $\boldsymbol{y}$. Let's assume we have the following prior and likelihood:

$$
\begin{aligned}
p(\boldsymbol{x}) &= \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x) \\
p(\boldsymbol{y} \mid \boldsymbol{x}) &= \mathcal{N}(\boldsymbol{y} \mid \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}, \boldsymbol{\Sigma}_y)
\end{aligned} \tag{4.124}
$$

which is an example of a **linear Gaussian system** $\boldsymbol{x} \rightarrow \boldsymbol{y}$.

The Wishart Distribution (4.5). We now dive into the distributions over the *parameters* $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$. First, we need some math prereqs out of the way. The **Wishart** is the generalization of the Gamma to pd matrices:

$$
\mathrm{Wi}(\boldsymbol{\Lambda} \mid \boldsymbol{S}, \nu) = \frac{1}{Z_{Wi}}|\boldsymbol{\Lambda}|^{(\nu-D-1)/2} \exp\left\{-\frac{1}{2}tr\left(\boldsymbol{\Lambda}\boldsymbol{S}^{-1}\right)\right\} \tag{257}
$$

$$
Z_{Wi} = 2^{\nu D/2}\Gamma_D(\nu/2)|\boldsymbol{S}|^{\nu/2} \tag{258}
$$

where

- ν: degrees of freedom
- $\boldsymbol{S}$: scale matrix (a.k.a. scatter matrix). Basically empirical Σ.
- Γ_D: multivariate gamma function

A nice property is that if $\boldsymbol{\Sigma}^{-1} \sim Wi(\boldsymbol{X}, \nu)$, then $\boldsymbol{S}igma \sim \mathrm{IW}(\boldsymbol{S}^{-1}, \nu + D + 1)$, the **inverse Wishart** (multi-dim generalization of inv Gamma).

Bayesian Statistics (Ch. 5)

Written by Brandon McKinzie

Kevin P. Murphy (2012). Bayesian Statistics.
Machine Learning: A Probabilistic Perspective.

MAP Estimation (5.2.1). The most popular **point estimate** for parameters $\boldsymbol{\theta}$ is the posterior mode, aka the **MAP estimate**. However, there are many drawbacks:

- No measure of uncertainty (true for any point estimate).
- Using θ_{MAP} for predictions is prone to overfitting.
- The mode is an atypical point.
- It's not invariant to reparameterization. Say two possible parameterizations $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2{=}f(\boldsymbol{\theta}_1)$, where f is some deterministic function. In general, it is **not** the case that $\hat{\boldsymbol{\theta}}_2 = f(\hat{\boldsymbol{\theta}}_1)$ under MAP.

Bayesian Model Selection (5.3). A model selection technique where we compute the best model m for data $\mathcal{D}$ using the formulas,

$$p(m \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid m)p(m)}{\sum_{m \in \mathcal{M}} p(\mathcal{D} \mid m)p(m)} \tag{259}$$

$$p(\mathcal{D} \mid m) = \int p(\mathcal{D} \mid \boldsymbol{\theta})p(\boldsymbol{\theta} \mid m)\mathrm{d}\boldsymbol{\theta} \tag{260}$$

where the latter is the **marginal likelihood**[56] for model m. Note that this isn't anything new; we've usually just denoted it simply as $p(\mathcal{D})$, since typically m is specified beforehand. Although large models with many parameters can achieve higher likelihoods under MLE/MAP, $p(\mathcal{D} \mid \hat{\boldsymbol{\theta}}_m)$, this is *not* necessarily the case with *marginal likelihood*, an effect known as the **Bayesian Occam's razor**. Below we give the marginal likelihoods for familiar models:

- Beta-Binomial:

$$p(\mathcal{D} \mid m{=}\text{BetaBinom}) = \binom{N}{N_1} \frac{B(a + N_1, b + N_0)}{B(a, b)}$$

  where B is the Beta function.
- Dirichlet-Multinoulli:

$$p(\mathcal{D}) = \frac{B(\boldsymbol{N} + \boldsymbol{\alpha})}{\boldsymbol{\alpha}}$$

[56] Also called the integrated likelihood or the evidence.

BIC Approximation (5.3.2.4). The integral involved in computing $p(\mathcal{D} \mid m)$ (henceforth denoted simply as $p(\mathcal{D})$) can be intractable. The **Bayesian information criterion** (BIC) is a popular approximation:

$$\text{BIC} \triangleq \log p(\mathcal{D} \mid \hat{\boldsymbol{\theta}}) - \frac{1}{2}\text{dof}(\hat{\boldsymbol{\theta}}) \log N \approx \log p(\mathcal{D}) \tag{261}$$

where

- $\text{dof}(\hat{\boldsymbol{\theta}})$ is the number of **degrees of freedom** in the model.
- $\hat{\boldsymbol{\theta}}$ is the MLE for the model.

BIC is also closely related to the **minimum description length** (MDL) principle and the **Akaike information criterion** (AIC).

Hierarchical Bayes (5.5). When defining our prior $p(\boldsymbol{\theta} \mid \boldsymbol{\eta})$, we have to of course specify the hyperparameters $\boldsymbol{\eta}$ required by our choice of prior. The Bayesian approach for doing this is to put a prior on our prior! This situation can be represented as a directed graphical model, illustrated below.

$$\begin{tikzcd} \eta \arrow[r] & \theta \arrow[r] & \mathcal{D} \end{tikzcd}$$

This is an example of a **hierarchical Bayesian model**, also called a **multi-level model**.

Bayesian Decision Theory (5.7). Decision problems can be cast as games against nature, where natures selects a quantity $y \in \mathcal{Y}$ unknown to us, and then generates an observation $\boldsymbol{x} \in \mathcal{X}$ that we get to see. Our goal is to devise a **decision procedure** or **policy** $\delta : \mathcal{X} \mapsto \mathcal{A}$ for generating an action $a \in \mathcal{A}$ from observation $\boldsymbol{x}$ that's deemed most compatible with the hidden state y. We define "compatible" via a loss function $L(y, a)$:

$$\delta(\boldsymbol{x}) = \arg\min_{a \in \mathcal{A}} \rho(a \mid \boldsymbol{x}) \tag{262}$$

$$\text{where} \quad \rho(a \mid \boldsymbol{x}) = \mathbb{E}_{p(y|\boldsymbol{x})}\left[L(y, a)\right] \tag{263}$$

In this context, we call ρ the **posterior expected loss**, and $\delta(x)$ the **Bayes estimator**. Some Bayes estimators for common loss functions are given below.

- **0-1 loss**: $L(y, a) = \mathbb{1}\{y \neq a\}$. Easy to show that $\delta(x) = \arg\max_{y \in \mathcal{Y}} p(y \mid \boldsymbol{x})$.

Machine Learning: A Probabilistic Perspective **January 25, 2020**

Frequentist Statistics (Ch. 6)

Table of Contents Local *Written by Brandon McKinzie*

Kevin P. Murphy (2012). Bayesian Statistics.
Machine Learning: A Probabilistic Perspective.

Sampling Distribution of an Estimator (6.2). In frequentist statistics, a **parameter estimate** $\hat{\theta}$ is computed by applying an **estimator** δ to some data[set] $\mathcal{D}$:

$$\hat{\theta} = \delta(\mathcal{D}) \tag{264}$$

> *The parameter is viewed as fixed and the data as random, which is the exact opposite of the Bayesian approach.*

The uncertainty in the parameter estimate can be measured by computing the **sampling distribution** of the estimator. The author's wording contradicts himself here[57], but basically the sampling distribution, $p(\hat{\theta})$, defines the distribution of $\hat{\theta}$ if we had access to $S \to \infty$ different datasets $\mathcal{D}^{(s)} = \{x_i^{(s)} \sim p_{\theta^*}\}$ (we often refer to p_{θ^*} as the true data-generating distribution).

sampling distribution

Frequentist Decision Theory (6.3). The expected loss or **risk** of an estimator δ is defined as

$$R(\theta^*, \delta) \triangleq \mathbb{E}_{\widetilde{\mathcal{D}} \sim p_{\theta^*}} \left[L\left(\theta^*, \delta(\widetilde{\mathcal{D}})\right)\right] \tag{265}$$

Desirable Properties of Estimators (6.4). The **bias** of an estimator is defined as

$$\text{bias}(\hat{\theta}(\cdot)) = \mathbb{E}_{\mathcal{D} \sim p_{\theta^*}} \left[\hat{\theta}(\mathcal{D}) - \theta^*\right] \tag{266}$$

[57] Seems to start using "estimator" and "parameter estimate" interchangeably"

Consider the case where our loss is the quadratic loss $L(\theta^*, \hat{\theta}) = (\theta^* - \hat{\theta})^2$. Then, by definition, the risk is just the MSE. The **bias-variance tradeoff** is a useful decomposition of the MSE[58]:

$$\begin{aligned}
\mathbb{E}\left[(\hat{\theta} - \theta^*)^2\right] &= \mathbb{E}\left[\left[\left(\hat{\theta} - \bar{\theta}\right) + \left(\bar{\theta} - \theta^*\right)\right]^2\right] && (267)\\
&= \mathbb{E}\left[(\hat{\theta} - \bar{\theta})^2\right] + (\bar{\theta} - \theta^*)^2 && (268)\\
&= \text{Var}\left[\hat{\theta}\right] + \text{Bias}^2(\hat{\theta}) && (269)
\end{aligned}$$

where $\bar{\theta} = \mathbb{E}\left[\hat{\theta}\right]$.

Estimating a Gaussian Mean (6.4.4.1)

Here I'll derive a lot that the author took as assumed knowledge. We want to estimate the mean of a Gaussian from N data point $\boldsymbol{x} = (x_1, \ldots, x_N)$. We assume the data is sampled from $x_i \sim \mathcal{N}\left(\theta^*, \sigma^2\right)$.

Derive the MLE of θ^*. The MLE $\hat{\theta}$ given data is defined as

$$\begin{aligned}
\hat{\theta} &= \arg\max_{\theta} \frac{1}{N} \sum_{i}^{N} \log \mathcal{N}\left(x_i; \theta, \sigma^2\right) && (270)\\
&= \arg\max_{\theta} \frac{1}{N} \sum_{i}^{N} -\frac{1}{2\sigma^2}(x_i - \theta)^2 && (271)\\
\frac{\partial}{\partial \theta}(\cdots) &= \frac{1}{N} \sum_{i}^{N} \frac{1}{\sigma^2}(x_i - \theta) = 0 && (272)\\
\therefore \hat{\theta} &= \frac{1}{N} \sum_{i}^{N} x_i = \bar{x} && (273)
\end{aligned}$$

Show that $\mathbb{E}\left[\hat{\theta}\right] = \theta^*$.

$$\mathbb{E}_{x_1,\ldots,x_N}\left[\hat{\theta}\right] = \mathbb{E}_{x_1,\ldots,x_N}\left[\frac{1}{N}\sum_i x_i\right] = \frac{1}{N}\sum_i \mathbb{E}\left[x_i\right] = \frac{1}{N}(N\theta^*) = \theta^* \quad (274)$$

Show that $\text{Var}\left[\hat{\theta}\right] = \frac{\sigma^2}{N}$. Since each x_i is sampled independently,

$$\text{Var}\left[\hat{\theta}\right] = \text{Var}\left[\frac{1}{N}\sum_i^N x_i\right] = \frac{1}{N^2}\sum_i \text{Var}\left[x_i\right] = \frac{1}{N^2}N\sigma^2 = \frac{\sigma^2}{N} \quad (275)$$

Therefore, the bias-variance decomposition for $\hat{\theta}$ is $MSE = \frac{\sigma^2}{N} + 0^2 = \frac{\sigma^2}{N}$.

[58] All expectations are taken wrt data-generating distribution, $\mathcal{D} \sim p(\mathcal{D}; \theta^*)$.

Machine Learning: A Probabilistic Perspective **August 19, 2018**

Linear Regression (Ch. 7)

Table of Contents Local *Written by Brandon McKinzie*

Kevin P. Murphy (2012). Linear Regression.
Machine Learning: A Probabilistic Perspective.

Model Specification (7.2). The linear regression model is a model of the form

$$p(y \mid \boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(y \mid \boldsymbol{w}^T \boldsymbol{x}, \sigma^2) \tag{276}$$

Maximum Likelihood Estimation (least squares) (7.3). Most commonly, we'll estimate the parameters by computing the MLE, defined by

$$\begin{aligned} \hat{\boldsymbol{\theta}} &\triangleq \arg\max_{\theta} \log p(\mathcal{D} \mid \boldsymbol{\theta}) && (277) \\ &= \arg\min_{\theta} \left[-\log p(\mathcal{D} \mid \boldsymbol{\theta}) \right] && (278) \\ &= -\frac{1}{2\sigma^2} RSS(\boldsymbol{w}) - \frac{N}{2} \log(2\pi\sigma^2) && (279) \\ RSS(\boldsymbol{w}) &\triangleq \sum_{i=1}^{N} (y_i - \boldsymbol{w}^T \boldsymbol{x}_i)^2 && (280) \end{aligned}$$

where $RSS(\boldsymbol{w})$ is the **residual sum of squares**. Notice that $\boldsymbol{\theta} := (\boldsymbol{w}, \sigma^2)$, but typically we're focused on estimating $\boldsymbol{w}$[59].

Derivation of the MLE (7.3.1). We'll now denote the negative log likelihood as $\mathrm{NLL}(\boldsymbol{w})$ and drop constant terms that are irrelevant for the optimization task.

$$\mathrm{NLL}(\boldsymbol{w}) = \frac{1}{2} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}||_2^2 = \frac{1}{2} \boldsymbol{w}^T (\boldsymbol{X}^T \boldsymbol{X}) \boldsymbol{w} - \boldsymbol{w}^T (\boldsymbol{X}^T \boldsymbol{y}) \tag{281}$$

where
$$\boldsymbol{X}^T \boldsymbol{X} = \sum_{i=1}^{N} \boldsymbol{x}_i \boldsymbol{x}_i^T \tag{282}$$

$$\boldsymbol{X}^T \boldsymbol{y} = \sum_{i=1}^{N} \boldsymbol{x}_i y_i \tag{283}$$

[59]Since our goal is typically to make future predictions $\hat{y}(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}$, rather than sampling $y \sim p(y \mid \boldsymbol{x}, \boldsymbol{\theta})$, we aren't concerned with estimating σ^2. We assume some variability, and the goal is focused on fitting the data to a straight line.

And the optimal $\hat{\boldsymbol{w}}_{OLS}$ be found by taking the gradient, setting to zero, and solving for $\boldsymbol{w}$ as usual:

$$\nabla \text{NLL}(\boldsymbol{w}) = \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^T\boldsymbol{y} \tag{284}$$

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{y} \qquad \text{[normal eq.]} \tag{285}$$

$$\hat{\boldsymbol{w}}_{OLS} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{286}$$

Geometric Interpretation (7.3.2). Use the column-vector representation of $\boldsymbol{X} \in \mathbb{R}^{N \times D}$, where we assume $N > D$[60]. Our prediction can then be written

$$\hat{y} = \boldsymbol{X}\boldsymbol{w} = w_1\tilde{\boldsymbol{x}}_1 + \cdots + w_D\tilde{\boldsymbol{x}}_D \tag{287}$$

i.e. a linear combination of the D column vectors $\tilde{\boldsymbol{x}}_i \in \mathbb{R}^N$. Crucially, observe that this means $\hat{y} \in \text{span}(\{\tilde{\boldsymbol{x}}\}_i^D)$ no matter what (a hard constraint by definition of our model). So, how do you minimize the residual norm $||\boldsymbol{y} - \hat{\boldsymbol{y}}||$ given that $\hat{\boldsymbol{y}}$ is restricted to a particular subspace? You require $\boldsymbol{y} - \hat{\boldsymbol{y}}$ to be orthogonal to that subspace, of course[61]! Formally, this means $\tilde{\boldsymbol{x}}_j^T(\boldsymbol{y} - \hat{\boldsymbol{y}}) = 0$, for all $1 \le j \le D$. Equivalently,

$$\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) = \boldsymbol{0} \quad \Longrightarrow \quad \hat{\boldsymbol{w}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{288}$$

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{w}} = \boldsymbol{P}\boldsymbol{y} \tag{289}$$

$$\text{where} \qquad \boldsymbol{P} \triangleq \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T \tag{290}$$

Although this neat, I'm left unsatisfied since there appears to be no intuition of what the column vectors of $\boldsymbol{X}$ really mean on a conceptual level.

Robust Linear Regression (7.4). Gaussians sensitive to outliers, since their log-likelihood penalizes deviations quadratically[62]. One way to achieve **robustness** to outliers is to instead use a distribution with **heavy tails**, such that they still allow for higher likelihoods of outliers, but they don't need to shift the whole distribution around to accommodate for them. One popular choice is the **Laplace distribution**,

Replacing Gaussian with heavy-tailed dists.

$$p(y \mid \boldsymbol{x}, \boldsymbol{w}, b) = \text{Lap}(y \mid \boldsymbol{w}^T\boldsymbol{x}, b) \triangleq \frac{1}{2b}\exp\left\{-\frac{1}{b}|y - \boldsymbol{w}^T\boldsymbol{x}|\right\} \tag{291}$$

$$NLL(\boldsymbol{w}) = \sum_i |y_i - \boldsymbol{w}^T\boldsymbol{x}_i| \tag{292}$$

[60]This means we have more rows than columns, which means our column space cannot span all of $\mathbb{R}^N$.

[61]Consider that $\boldsymbol{y} - \hat{\boldsymbol{y}}$ points *from* our prediction (which is in the constraint subspace) $\hat{\boldsymbol{y}}$ *to* the true $\boldsymbol{y}$ that we want to get closest to. Intuitively that means $\hat{\boldsymbol{y}}$ is looking "straight out" at $\boldsymbol{y}$, in a direction orthogonal to the subspace that $\hat{\boldsymbol{y}}$ lives in. Formally, we can write $\boldsymbol{y} = (\boldsymbol{y}_\parallel, \boldsymbol{y}_\perp)$, where $\boldsymbol{y}_\parallel$ is the component within Col($\boldsymbol{x}$). The best we can do, then, is $\hat{\boldsymbol{y}} := \boldsymbol{y}_\parallel$.

[62]In other words, outliers initially get huge loss values, and the distribution shifts toward them to minimize loss (undesirably).

Goal: convert NLL to a form easier to optimize (linear). Let $r_i \triangleq y_i - \boldsymbol{w}^T \boldsymbol{x}_i$ be the i'th residual. The following steps show how we can convert this into a **linear program**:

$$r_i \triangleq r_i^+ - r_i^- \qquad (r_i^+ \geq 0)(r_i^- \geq 0) \tag{293}$$

$$\min_{\boldsymbol{w}, \boldsymbol{r}^+, \boldsymbol{r}^-} \sum_i (r_i^+ + r_i^-) \quad \text{s.t.} \quad \boldsymbol{w}^T \boldsymbol{x}_i + r_i^+ - r_i^- = y_i \tag{294}$$

$$\min_{\boldsymbol{\theta}} \boldsymbol{f}^T \boldsymbol{\theta} \quad \text{s.t.} \quad \boldsymbol{A\theta} \leq \boldsymbol{b},\ \boldsymbol{A}_{eq}\boldsymbol{\theta} = \boldsymbol{b}_{eq},\ \boldsymbol{1} \leq \boldsymbol{\theta} \leq \boldsymbol{u} \tag{295}$$

where the last equation is the **standard form** of a LP.

Ridge Regression (7.5). We know that MLE can overfit by essentially memorizing the data. If, for example, we model 21 points with a degree-14 polynomial[63], we get many large positive and negative numbers for our learned coefficients, which allow the curve to wiggle in just the right way to almost perfectly interpolate the data – **this is why we often regularize weights to have low absolute value**. This encourages smoother/less-wiggly curves. One way to do this is by using a zero-mean Gaussian prior on our weights:

Doing MAP instead of MLE

$$p(\boldsymbol{w}) = \prod_j \mathcal{N}(w_j \mid 0, \tau^2) \tag{296}$$

This makes our MAP estimation problem and solution take the form

Note that w_0 is NOT regularized.

$$J(\boldsymbol{w}) = \frac{1}{N} \sum_i^N \left(y_i - \boldsymbol{w}^T \boldsymbol{x}_i - w_0 \right) + \lambda ||\boldsymbol{w}||_2^2 \tag{297}$$

$$\hat{\boldsymbol{w}}_{ridge} = (\boldsymbol{X}^T \boldsymbol{X} + \lambda \boldsymbol{I}_D)^{-1} \boldsymbol{X}^T \boldsymbol{y} \tag{298}$$

[63]To review polynomials, search "Lagrange interpolation" in your CS 70 notes.

Machine Learning: A Probabilistic Perspective **August 11, 2018**

Generalized Linear Models and the Exponential Family (Ch. 9)

Table of Contents Local *Written by Brandon McKinzie*

Kevin P. Murphy (2012). Generalized Linear Models and the Exponential Family.
Machine Learning: A Probabilistic Perspective.

The Exponential Family (9.2). Why is the exponential family important?

- It's the only family with **finite-sized sufficient statistics**[64].
- It's the only family for which **conjugate priors exist**.
- It makes the **least set of assumptions** subject to some user-chosen constraints.
- It's at the core of GLMs and variational inference.

A pdf or pmf $p(\boldsymbol{x} \mid \boldsymbol{\theta})$, for $\boldsymbol{x} \in \mathcal{X}^m$ and $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^d$, is said to be in the ***exponential family*** *if it's of the form*

$$p(\boldsymbol{x} \mid \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} h(\boldsymbol{x}) \exp\{\boldsymbol{\theta}^T \boldsymbol{\phi}(\boldsymbol{x})\} = h(\boldsymbol{x}) \exp\left\{\boldsymbol{\theta}^T \boldsymbol{\phi}(\boldsymbol{x}) - A(\boldsymbol{\theta})\right\} \tag{299}$$

$$Z(\boldsymbol{\theta}) = \int_{\mathcal{X}^m} h(\boldsymbol{x}) \exp\{\boldsymbol{\theta}^T \boldsymbol{\phi}(\boldsymbol{x})\} \tag{300}$$

$$A(\boldsymbol{\theta}) = \log Z(\boldsymbol{\theta}) \tag{301}$$

$h(\boldsymbol{x})$ is a scaling constant, often 1.

where $\boldsymbol{\theta}$ are the ***natural (canonical) parameters***[65]*, and $\boldsymbol{\phi}(\boldsymbol{x})$ are the* ***sufficient statistics****.*

Below are some (quick/condensed) examples showing the first couple steps in rewriting familiar distributions in exponential family form:

$$\textbf{[Bernoulli]} \quad \text{Ber}(x \mid \mu) = \mu^x (1-\mu)^{1-x} = \exp\{x \log \mu + (1-x)\log(1-\mu)\} \tag{302}$$

$$\textbf{[Multinoulli]} \quad \text{Cat}(x \mid \boldsymbol{\mu}) = \prod_k^K \mu_k^{x_k} = \exp\left\{\sum_k^{K-1} x_k \log\left(\mu_k/\mu_K\right) + \log \mu_K\right\} \tag{303}$$

[64] Given certain regularity conditions.

[65] We often generalize this with $\boldsymbol{\eta}(\boldsymbol{\theta})$, which maps whatever params $\boldsymbol{\theta}$ we've chosen to the canonical params $\boldsymbol{\eta}(\boldsymbol{\theta})$.

Log Partition Function (9.2.3). The derivatives of the log partition, $A(\boldsymbol{\theta})$, can be used to generate **cumulants**[66] for the sufficient statistics, $\boldsymbol{\phi}(\boldsymbol{x})$.

$$\frac{\partial A(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \mathbb{E}\left[\boldsymbol{\phi}(\boldsymbol{x})\right] \tag{304}$$

$$\nabla^2 A(\boldsymbol{\theta}) = \mathrm{cov}\left[\boldsymbol{\phi}(\boldsymbol{x})\right] \tag{305}$$

MLE for the Exponential Family (9.2.4). The likelihood takes the form

$$p(\mathcal{D} \mid \boldsymbol{\theta}) = \left[\prod_i^N h(\boldsymbol{x}^{(i)})\right] g(\boldsymbol{\theta})^N \exp\left\{\boldsymbol{\eta}(\boldsymbol{\theta})^T \boldsymbol{\phi}(\mathcal{D})\right\} \tag{306}$$

It appears that we are denoting $1/Z$ with g now.

$$\boldsymbol{\phi}(\mathcal{D}) = \sum_i^N \boldsymbol{\phi}(\boldsymbol{x}^{(i)}) = \begin{bmatrix}\sum_i \phi_1 & \cdots & \sum_i \phi_K\end{bmatrix} \tag{307}$$

where I've denoted $\sum_{i=1}^N \phi_k(\boldsymbol{x}^{(i)})$ as simply $\sum_i \phi_k$. The **Pitman-Koopman-Darmois theorem** states, given certain regularity conditions/constraints[67], that the exponential family is the only family with *finite sufficient statistics* (dimensionality independent of the size of the data set). For example, in the above formula, we have $K+1$ sufficient statistics ($+1$ since we need to know the value of N).

Consider a canonical[68] exponential family which also sets $h(\cdot) = 1$. The log-likelihood is

$$p(\mathcal{D} \mid \boldsymbol{\theta}) = \boldsymbol{\theta}^T \boldsymbol{\phi}(\mathcal{D}) - N A(\boldsymbol{\theta}) \tag{308}$$

Since $-A(\boldsymbol{\theta})$ is concave[69] and the other term is linear in $\boldsymbol{\theta}$, the log-likelihood is concave and thus has a global maximum.

[66] The first and second cumulants are mean and variance.

[67] The wording is weird here. We mean "out of all families/distributions that *already* satisfy certain constraints that must be met, the exponential family is the only...". For example, the uniform distribution has finite statistics and is not in the exponential family, but it does not meet the constraint that its support must be independent of the parameters, so it's outside the scope of the theorem.

[68] Defined as those which satisfy $\boldsymbol{\eta}(\boldsymbol{\theta}) = \boldsymbol{\theta}$.

[69] We know $-A$ is concave because A is convex. We know A is convex because $\nabla^2 A$ is positive definite. Remember that any twice-differentiable multivariate function f is convex IFF its Hessian is pd for all $\boldsymbol{\theta}$. See sec 7.3.3 and 9.2.3 for more.

Maximum Entropy Derivation of the Exponential Family (9.2.6). Suppose we don't know which distribution p to use, but we do know the expected values of certain features or functions:

$$F_k \triangleq \mathbb{E}_{\boldsymbol{x} \sim p(\mathbf{x})}\left[f_k(\boldsymbol{x})\right] = \sum_{\boldsymbol{x}} f_k(\boldsymbol{x}) p(\boldsymbol{x}) \tag{309}$$

The principle of **maximum entropy** or **maxent** says we should pick the distribution with maximum entropy, subject to the constraints that the moments of the distribution match the empirical moments of the specified functions $f_k(\boldsymbol{x})$. Treating p as a fixed length vector (i.e. assuming $\boldsymbol{x}$ is discrete), we can take the derivative of our Lagrangian (entropy in units of nats with constraints) w.r.t. each "element" $p_x = p(\boldsymbol{x})$ to find the optimal distribution.

$$J(p, \boldsymbol{\lambda}) = H(p) + \lambda_0 \left(1 - \sum_{\boldsymbol{x}} p(\boldsymbol{x})\right) + \sum_k \lambda_k \left(F_k - \sum_{\boldsymbol{x}} p(\boldsymbol{x}) f_k(\boldsymbol{x})\right) \tag{310}$$

$$\frac{\partial J}{\partial p(\boldsymbol{x})} = -1 - \log p(\boldsymbol{x}) - \lambda_0 - \sum_k \lambda_k f_k(\boldsymbol{x}) \tag{311}$$

Setting this derivative to zero yields

$$p(\boldsymbol{x}) = \frac{1}{Z} \exp\left\{-\sum_k \lambda_k f_k(\boldsymbol{x})\right\} \tag{312}$$

Thus the maxent distribution $p(\boldsymbol{x})$ has the form of the exponential family, a.k.a. the **Gibbs distribution**.

Machine Learning: A Probabilistic Perspective **August 26, 2018**

Mixture Models and the EM Algorithm (Ch. 11)

Table of Contents Local *Written by Brandon McKinzie*

Kevin P. Murphy (2012). Mixture Models and the EM Algorithm.
Machine Learning: A Probabilistic Perspective.

Latent Variable Models (LVMs) (11.1). In this chapter, we explore directed GMs that have hidden/latent variables. Advantages of LVMs:

1. Often have fewer params.
2. Hidden vars can serve as a bottleneck (representation learning).

Mixture Models (11.2). The simplest form of LVM is where the hidden variables $z_i \in \{1, \ldots, K\}$ represent a discrete latent state. We use discrete prior $p(z_i) = \text{Cat}(\boldsymbol{\pi}) = \pi_i$, and likelihood $p(\boldsymbol{x}_i \mid z{=}k) = p_k(\boldsymbol{x}_i)$. A **mixture model** is defined by

$$p(\boldsymbol{x}_i \mid \boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k p_k(\boldsymbol{x}_i \mid \boldsymbol{\theta}) \tag{313}$$

We call p_k the kth **base distribution**.

Some popular mixture models:

- **Mixture of Gaussians** (11.2.1). Each $p_k = \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. Given large enough K, *a GMM can approximate any density defined on* $\mathbb{R}^D$.
- **Mixture of Multinoullis** (11.2.2). Let $\boldsymbol{x} \in \{0,1\}^D$. Each $p_k(\boldsymbol{x}) = \prod_j^D \text{Ber}(x_j \mid \mu_{jk})$.

Below, I derive the expectation and covariance of $\boldsymbol{x}$ in this case[70].

$$\mathbb{E}\left[\mathbf{x}\right] = \sum_{\boldsymbol{x}} \boldsymbol{x} p(\boldsymbol{x}) \tag{314}$$

$$= \sum_{\boldsymbol{x}} \boldsymbol{x} \sum_{k}^{K} \pi_k p_k(\boldsymbol{x}) \tag{315}$$

$$= \sum_{\boldsymbol{x}} \boldsymbol{x} \sum_{k}^{K} \pi_k \prod_{j}^{D} \mathrm{Ber}(x_j \mid \mu_{jk}) \tag{316}$$

$$= \sum_{k}^{D} \pi_k \sum_{x_1} \cdots \sum_{x_D} \boldsymbol{x} \prod_{j}^{D} \mathrm{Ber}(x_j \mid \mu_{jk}) \tag{317}$$

$$= \sum_{k}^{D} \pi_k \sum_{x_1} \mathrm{Ber}(x_1 \mid \mu_{1k}) \cdots \sum_{x_D} \mathrm{Ber}(x_D \mid \mu_{Dk}) \boldsymbol{x} \tag{318}$$

$$= \sum_{k}^{D} \pi_k \boldsymbol{\mu}_k \tag{319}$$

Next we want to find $\mathrm{cov}(\mathbf{x}) = \mathbb{E}\left[\mathbf{x}\mathbf{x}^T\right] - \mathbb{E}\left[\mathbf{x}\right]\mathbb{E}\left[\mathbf{x}\right]^T$. I think the insight that makes finding the first term easiest is realizing that you only need to find the two cases, $\mathbb{E}\left[x_i^2\right]$ and $\mathbb{E}\left[x_i x_{j\neq i}\right]$, where in this case

$$\mathbb{E}\left[x_i^2\right] = \sum_k \pi_k \mu_{ik} \tag{320}$$

$$\mathbb{E}\left[x_i x_{j\neq i}\right] = \sum_k \pi_k \mu_{ik}\mu_{jk} \tag{321}$$

$$\therefore \mathbb{E}\left[\mathbf{x}\mathbf{x}^T\right] = \sum_k \pi_k \left(\boldsymbol{\Sigma}_k + \boldsymbol{\mu}_k\boldsymbol{\mu}_k^T\right) \tag{322}$$

where $\Sigma_k = \mathrm{diag}(\mu_{jk}(1-\mu_{jk}))$ is the covariance of $\boldsymbol{x}$ under p_k. The fact that the mixture covariance matrix is now non-diagonal confirms that the mixture can capture correlations between variables $x_i, x_{j\neq i}$, unlike a single product-of-Bernoullis model.

The EM Algorithm (11.4). In LVMs, it's usually intractable to compute the MLE since we have to marginalize over hidden variables while satisfying constraints like positive definite covariance matrices, etc. The EM algorithm gets around these issues via a two-step process. The first step involves taking an expectation, where the expectation is over $\boldsymbol{z} \sim p(\boldsymbol{z} \mid \boldsymbol{x}, \theta^{t-1})$ for each individual observed $\boldsymbol{x}$, where we use the current parameter estimates when sampling $\boldsymbol{z}$ in the expectation. **This gives us an auxiliary likelihood that's a function of $\boldsymbol{\theta}$** which will serve as a stand-in (in the 2nd step) for what we typically use as the likelihood in MLE. The second step is then just finding the optimal $\boldsymbol{\theta}^t$ over the auxiliary likelihood function from the first step. This iterates until convergence or some stopping condition.

[70]Shown in excruciating detail because I was unable to work through this in my head alone.

Procedure: EM Algorithm

First, let's define our auxiliary function Q as

$$Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{t-1}) = \mathbb{E}_{p(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\theta}^{t-1})}\left[\ell_c(\boldsymbol{\theta}) \mid \mathcal{D}\right] \tag{323}$$

$$\text{where} \quad \ell_c(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)} \mid \boldsymbol{\theta}) \tag{324}$$

where, again, the "expectation" serves the purpose of determining the expected value of $\boldsymbol{z}^{(i)}$ for each observed $\boldsymbol{x}^{(i)}$. It's somewhat of a misnomer to denote the expectation like this, since each $\boldsymbol{z}^{(i)}$ is innately tied with its corresponding observation $\boldsymbol{x}^{(i)}$.

1. **E-Step**: Evaluate $Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{t-1})$ using (obviously) the previous parameters $\boldsymbol{\theta}^{t-1}$. This yields a function of $\boldsymbol{\theta}$. This gives us the expected log-likelihood function of $\boldsymbol{\theta}$ for the observed data $\mathcal{D}$.
2. **M-Step**: Optimize Q w.r.t $\boldsymbol{\theta}$ to get $\boldsymbol{\theta}^t$:

$$\boldsymbol{\theta}^t = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{t-1}) \tag{325}$$

Machine Learning: A Probabilistic Perspective **September 14, 2018**

Latent Linear Models (Ch. 12)

Table of Contents Local *Written by Brandon McKinzie*

Kevin P. Murphy (2012). Latent Linear Models.
Machine Learning: A Probabilistic Perspective.

Factor Analysis (12.1). Whereas mixture models define $p(z) = \text{Cat}(\boldsymbol{\pi})$ for a single hidden variable $z \in \{1, \ldots, K\}$, **factor analysis** begins by instead using a *vector* of *real-valued* latent variables, $\boldsymbol{z} \in \mathbb{R}^L$. The simplest prior is $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$. If $\boldsymbol{x} \in \mathbb{R}^D$, we can define

$$p(\boldsymbol{x}_i \mid \boldsymbol{z}_i, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{W}\boldsymbol{z}_i + \boldsymbol{\mu}, \boldsymbol{\Psi}) \tag{326}$$

where

- $\boldsymbol{W} \in \mathbb{R}^{D \times L}$: **factor loading matrix**.
- $\boldsymbol{\Psi} \in \mathbb{R}^{D \times D}$ is a $\overline{\text{diagonal}}$ covariance matrix, since we want to "force" $\boldsymbol{z}$ to explain the correlation[71]. If $\overline{\boldsymbol{\Psi} = \sigma^2 \boldsymbol{I}}$, we get **probabilistic PCA**.

Summaries of key points regarding FA:

- **Low-rank parameterization of MVN**. FA can be thought of as specifying $p(\boldsymbol{x})$ using a small number of parameters. [math] yields that

$$\text{cov}(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{W}^T + \boldsymbol{\Psi}$$

  which has $\mathcal{O}(LD)$ params (remember $\boldsymbol{\Psi}$ is diagonal) instead of the usual $\mathcal{O}(D^2)$.
- **Unidentifiability**: The params of an FA model are unidentifiable.

Classical PCA (12.2.1). Goal: find an orthogonal set of L linear basis vectors $\boldsymbol{w}_j \in \mathbb{R}^D$, and the scores $\boldsymbol{z}_i \in \mathbb{R}^L$, such that we minimize the average **reconstruction error**:

$$J(\boldsymbol{W}, \boldsymbol{Z}) = \frac{1}{N} \sum_{i=1}^{N} ||\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i||^2, \quad \text{where} \quad \hat{\boldsymbol{x}}_i := \boldsymbol{W}\boldsymbol{z}_i \tag{327}$$

$$= ||\boldsymbol{X} - \boldsymbol{W}\boldsymbol{Z}^T||_F^2 \tag{328}$$

where $\boldsymbol{W} \in \mathbb{R}^{D \times L}$ is orthonormal. **Solution**: assign each column $\boldsymbol{W}_{:,\ell}$ to the eigenvector with ℓ'th largest eigenvalue of $\hat{\boldsymbol{\Sigma}} = \frac{1}{N} \sum_i^N \boldsymbol{x}_i \boldsymbol{x}_i^T$, assuming $\mathbb{E}[\boldsymbol{x}] = \boldsymbol{0}$. Then we compute $\hat{\boldsymbol{z}}_i := \boldsymbol{W}^T \boldsymbol{x}_i$.

[71] It's easier to think of this graphically. Our model asserts that $\boldsymbol{x}_i \perp \boldsymbol{x}_{j \neq i} \mid \boldsymbol{z}$. Independence implies zero correlation, and we cement this by constraining $\boldsymbol{\Psi}$ to be diagonal. See your related note on LFMs, chapter 13 of the DL book.

Proof: PCA

Case: L=1. Goal: Find the best 1-d solution, $\boldsymbol{w} \in \mathbb{R}^D$, $z_i \in \mathbb{R}$, $\boldsymbol{z} \in \mathbb{R}^N$. Remember that $||\boldsymbol{w}||_2 = 1$.

$$J(\boldsymbol{w}, \boldsymbol{z}) = \frac{1}{N} \sum_i^N ||\boldsymbol{x}_i - z_i \boldsymbol{w}||^2 = \frac{1}{N} \left[\boldsymbol{x}_i^T \boldsymbol{x}_i - 2 z_i \boldsymbol{w}^T \boldsymbol{x}_i + z_i^2 \right] \tag{329}$$

$$\frac{\partial J}{\partial z_i} = 0 \quad \rightarrow \quad z_i = \boldsymbol{w}^T \boldsymbol{x}_i \tag{330}$$

$$J(\boldsymbol{w}) = \frac{1}{N} \sum_i^N \left[\boldsymbol{x}_i^T \boldsymbol{x}_i - z_i^2 \right] = \text{const} - \frac{1}{N} \sum_i^N z_i^2 \tag{331}$$

$$\therefore \arg\min_{\boldsymbol{w}} J(\boldsymbol{w}) = \arg\max_{\boldsymbol{w}} \text{Var}\left[\tilde{\boldsymbol{z}} \right] \tag{332}$$

This shows why PCA finds directions of maximal variance – aka the **analysis view** of PCA. Before finding the optimal $\boldsymbol{w}$, don't forget the Lagrange multipliers for constraining unit norm,

$$\tilde{J}(\boldsymbol{w}) = \boldsymbol{w}^T \hat{\boldsymbol{\Sigma}} \boldsymbol{w} + \lambda(\boldsymbol{w}^T \boldsymbol{w} - 1) \tag{333}$$

Singular Value Decomposition (SVD) (12.2.3). Any real $N \times D$ matrix $\boldsymbol{X}$ can be decomposed as

$$\boldsymbol{X} = \underbrace{\boldsymbol{U}}_{N \times N} \underbrace{\boldsymbol{S}}_{N \times D} \underbrace{\boldsymbol{V}^T}_{D \times D} \tag{334}$$

where the columns of $\boldsymbol{U}$ are the left singular vectors, and the columns of $\boldsymbol{V}$ are the right singular vectors. **Economy-sized SVD** will shrink $\boldsymbol{U}$ to be $N \times D$ and $\boldsymbol{S}$ to be $D \times D$ (we're assuming N > D).

Machine Learning: A Probabilistic Perspective **August 11, 2018**

Markov and Hidden Markov Models (Ch. 17)

Table of Contents Local *Written by Brandon McKinzie*

Kevin P. Murphy (2012). Markov and Hidden Markov Models.
Machine Learning: A Probabilistic Perspective.

Hidden Markov Models (17.3). HMMs model a joint distribution over a sequence of T observations $\boldsymbol{x}_{\langle 1 \ldots T\rangle}$ and hidden states $\boldsymbol{z}_{\langle 1 \ldots T\rangle}$,

$$p(\boldsymbol{z}_{\langle 1 \ldots T\rangle}, \boldsymbol{x}_{\langle 1 \ldots T\rangle}) = p(\boldsymbol{z}_{\langle 1 \ldots T\rangle}) p(\boldsymbol{x}_{\langle 1 \ldots T\rangle} \mid \boldsymbol{z}_{\langle 1 \ldots T\rangle}) = \left[p(z_1) \prod_{t=2}^{T} p(z_t \mid z_{t-1}) \right] \left[\prod_{t=1}^{T} p(\boldsymbol{x}_t \mid z_t) \right] \tag{335}$$

where each hidden state is discrete: $z_t \in \{1, \ldots, K\}$, while each observation $\boldsymbol{x}_t$ can be discrete or continuous.

The Forwards Algorithm (17.4.2). Goal: compute the filtered[72] marginals $p(z_t \mid \boldsymbol{x}_{\langle 1 \ldots t\rangle})$.

1. **Prediction step**. Compute the *one-step-ahead predictive density* $p(z_t \mid \boldsymbol{x}_{\langle 1 \ldots t-1\rangle})$,

$$p(z_t{=}j \mid \boldsymbol{x}_{\langle 1 \ldots t-1\rangle}) = \sum_i p(z_t = j, z_{t-1} = i \mid \boldsymbol{x}_{\langle 1 \ldots t-1\rangle}) \tag{336}$$

   which will serve as our prior for time t (since it does not take into account observed data at time t).

2. **Update step**. We "update" our beliefs by observing $\boldsymbol{x}_t$,

$$\begin{aligned}
\alpha_t(j) &\triangleq p(z_t{=}j \mid \boldsymbol{x}_{\langle 1 \ldots t\rangle}) &(337)\\
&= \frac{p(z_t{=}j, \boldsymbol{x}_t, \boldsymbol{x}_{\langle 1 \ldots t-1\rangle})}{p(\boldsymbol{x}_{\langle 1 \ldots t\rangle})} &(338)\\
&= \frac{\cancel{p(\boldsymbol{x}_{\langle 1 \ldots t-1\rangle})} p(z_t{=}j \mid \boldsymbol{x}_{\langle 1 \ldots t-1\rangle}) p(\boldsymbol{x}_t \mid z_t{=}j, \cancel{\boldsymbol{x}_{\langle 1 \ldots t-1\rangle}})}{\cancel{p(\boldsymbol{x}_{\langle 1 \ldots t-1\rangle})} p(\boldsymbol{x}_t \mid \boldsymbol{x}_{\langle 1 \ldots t-1\rangle})} &(339)\\
&= \frac{1}{Z_t} p(z_t{=}j \mid \boldsymbol{x}_{\langle 1 \ldots t-1\rangle}) p(\boldsymbol{x}_t \mid z_t{=}j) &(340)
\end{aligned}$$

Notice that we can also use the values of Z_t to compute the log probability of the evidence:

$$\log p(\boldsymbol{x}_{\langle 1 \ldots T\rangle} \mid \boldsymbol{\theta}) = \sum_{t=1}^{T} \log p(\boldsymbol{x}_t \mid \boldsymbol{x}_{\langle 1 \ldots t-1\rangle}) = \sum_{t=1}^{T} \log Z_t \tag{341}$$

[72] They're called "filtered" because they use all observations $\boldsymbol{x}_{\langle 1 \ldots t\rangle}$ instead of just $\boldsymbol{x}_t$, which reduces/filters out the noise more.

Forwards Algorithm (Algorithm 17.1)

We are given transition matrix $T_{i,j} = p(z_t = j \mid z_{t-1} = i)$, evidence vectors $\psi_t(j) = p(\boldsymbol{x}_t \mid z_t{=}j)$, and initial state distribution $\pi(j) = p(z_1 = j)$.

1. First, compute the initial $[\alpha_1, Z_1] = \text{normalize}(\psi_1 \odot \pi)$.
2. For time $2 \le t \le T$, compute $[\alpha_t, Z_t] = \text{normalize}(\psi_t \odot \boldsymbol{T}^T \boldsymbol{\alpha}_{t-1})$.
3. Return $\boldsymbol{\alpha}_{\langle 1\dots T\rangle}$ and $\log p(\boldsymbol{x}_{\langle 1\dots T\rangle}) = \sum_t \log Z_t$.

The Forwards-Backwards Algorithm (17.4.3). Goal: compute the smoothed marginals, $p(z_t = j \mid \boldsymbol{x}_{\langle 1\dots T\rangle})$.

$$\begin{aligned}
\gamma_t(j) &\triangleq p(z_t = j \mid \boldsymbol{x}_{\langle 1\dots T\rangle}) && (342)\\
&\propto p(z_t = j \mid \boldsymbol{x}_{\langle 1\dots t\rangle}) p(\boldsymbol{x}_{\langle t+1\dots T\rangle} \mid z_t = j) && (343)\\
&= \alpha_t(j)\beta_t(j) && (344)\\
\text{where} \quad \beta_t(j) &\triangleq p(\boldsymbol{x}_{\langle t+1\dots T\rangle} \mid z_t{=}j) && (345)\\
&= p(\boldsymbol{x}_{t+1}, \boldsymbol{x}_{\langle t+2\dots T\rangle} \mid z_t{=}j) && (346)\\
&= \sum_i p(z_{t+1}{=}i, \boldsymbol{x}_{t+1}, \boldsymbol{x}_{\langle t+2\dots T\rangle} \mid z_t{=}j) && (347)\\
&= \sum_i p(z_{t+1}{=}i, \boldsymbol{x}_{t+1} \mid z_t{=}j) p(\boldsymbol{x}_{\langle t+2\dots T\rangle} \mid \cancel{z_t{=}j}, z_{t+1}{=}i, \cancel{\boldsymbol{x}_{t+1}}) && (348)\\
&= \sum_i p(z_{t+1}{=}i \mid z_t{=}j) p(\boldsymbol{x}_{t+1} \mid z_{t+1}{=}i) p(\boldsymbol{x}_{\langle t+2\dots T\rangle} \mid z_{t+1}{=}i) && (349)\\
&= \sum_i p(z_{t+1}{=}i \mid z_t{=}j) p(\boldsymbol{x}_{t+1} \mid z_{t+1}{=}i) \beta_{t+1}(i) && (350)
\end{aligned}$$

Using the same notation as Algorithm 17.1 above, the matrix-vector form for β is

$$\boldsymbol{\beta}_t = \boldsymbol{T}(\boldsymbol{\psi}_t \odot \boldsymbol{\beta}_{t+1}) \tag{351}$$

with base case $\boldsymbol{\beta}_T = \mathbf{1}$.

The Viterbi Algorithm (17.4.4). Denote the [probability for] the most probable path leading to $z_t{=}j$ as

$$\begin{aligned}
\delta_t(j) &\triangleq \max_{\boldsymbol{z}_{\langle 1\dots t-1\rangle}} p(\boldsymbol{z}_{\langle 1\dots t-1\rangle}, z_t{=}j \mid \boldsymbol{x}_{\langle 1\dots t\rangle}) && (352)\\
&= \max_i \delta_{t-1}(i) \cdot T_{i,j} \cdot \psi_t(j) && (353)
\end{aligned}$$

It is common to work in the log-domain when computing δ.

with initialization of $\delta_1(j) = \pi_j \psi_1(j)$. We compute this until termination at $z_T^* = \arg\max_i \delta_T(i)$. Note the arg max here instead of a max – we keep track of both for all time steps. We do this so we can perform **traceback** to get the full most probable state sequence, starting at T and ending at $t = 1$:

$$z_t^* = a_{t+1}(z_{t+1}^*) \tag{354}$$

where $a_t(j)$, the most probable state at time $t-1$ leading to state j at time t, is the same formula as $\delta_t(j)$ but with an arg max.

Machine Learning: A Probabilistic Perspective **August 11, 2018**

Undirected Graphical Models (Ch. 19)

Table of Contents Local *Written by Brandon McKinzie*

Kevin P. Murphy (2012). Undirected Graphical Models.
Machine Learning: A Probabilistic Perspective.

Learning (19.5). Consider a MRF in log-linear form over C cliques and its log-likelihood (scaled by 1/N):

$$p(\boldsymbol{y} \mid \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\left\{ \sum_c \boldsymbol{\theta}_c^T \boldsymbol{\phi}_c(\boldsymbol{y}) \right\} \tag{355}$$

$$\ell(\boldsymbol{\theta}) = \frac{1}{N} \sum_i \left[\sum_c \boldsymbol{\theta}_c^T \boldsymbol{\phi}_c(\boldsymbol{y}_i) - \log Z(\boldsymbol{\theta}) \right] \tag{356}$$

We know from chapter 9 that this log-likelihood is concave in $\boldsymbol{\theta}$, and that $\frac{\partial}{\partial \boldsymbol{\theta}_c} \log Z = \mathbb{E}\left[\phi_c\right]$. So the gradient of the LL is

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}_c} = \left[\frac{1}{N} \sum_i \boldsymbol{\phi}_c(\boldsymbol{y}_i) \right] - \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{\theta})}\left[\boldsymbol{\phi}_c(\boldsymbol{y}) \right] \tag{357}$$

Note that the first ("clamped") term only needs to be computed once for any $\boldsymbol{y}_i$, it is completely independent of any parameters. It's just evaluating feature functions, which e.g. for CRFs are often all indicator functions.

CRF Training (19.6.3). For [linear-chain] CRFs, the equations change slightly (but importantly).

$$\ell(\boldsymbol{w}) \triangleq \frac{1}{N} \sum_i \left[\sum_c \boldsymbol{w}_c^T \boldsymbol{\phi}_c(\boldsymbol{y}_i, \boldsymbol{x}_i) - \log Z(\boldsymbol{w}, \boldsymbol{x}_i) \right] \tag{358}$$

$$\frac{\partial \ell}{\partial \boldsymbol{w}_c} = \frac{1}{N} \sum_i \left[\boldsymbol{\phi}_c(\boldsymbol{y}_i, \boldsymbol{x}_i) - \mathbb{E}_{p(\boldsymbol{y}|\boldsymbol{x}_i)}\left[\boldsymbol{\phi}_c(\boldsymbol{y}_i, \boldsymbol{x}_i) \right] \right] \tag{359}$$

It's important to recognize that the gradient of the log partition function now must be computed for *each* instance $\boldsymbol{x}_i$.

Convex Optimization

Contents

Convex Optimization **August 19, 2018**

Convex Sets (Ch. 2)

Table of Contents Local *Written by Brandon McKinzie*

Boyd and Vandenberghe (2004). Convex Sets.
Convex Optimization.

Lines and line segments

Viewed as a function of $\theta \in \mathbb{R}$, we can express the equation for a ***line*** *in $\mathbb{R}^n$ in the following two ways:*

$$y = \theta x_1 + (1-\theta)x_2 \tag{360}$$
$$y = x_2 + \theta(x_1 - x_2) \tag{361}$$

for some $x_1, x_2 \neq x_1 \in \mathbb{R}^n$. If we restrict $\theta \in [0,1]$, we have a ***line segment****.*

Affine sets

A set $C \subseteq \mathbb{R}^n$ is ***affine*** *if the line (not just segment) through any two distinct points in C also lies in C. More generally, this implies that for any set of points $\{x_1, \dots, x_k\}$, with each $x_i \in C$, all* ***affine combinations****,*

$$\sum_{i=1}^{k} \theta_i x_i, \quad \textit{where} \quad \sum_i \theta_i = 1 \tag{362}$$

are in C, too. Related terminology:

- ***affine hull*** *of any set $C \subseteq \mathbb{R}^n$, denoted $\textbf{aff}\, C$, is the set of all affine combinations of points in C.*
- ***affine dimension*** *of a set C is the dimension of its affine hull.*
- ***relative interior*** *of a set C, denoted $\textbf{relint}\, C$, is its interior[73] relative to $\textbf{aff}\, C$,*

$$\textbf{relint}\, C \triangleq \{x \in C \mid B(x,r) \cap \textbf{aff}\, C \subseteq C \textit{ for some } r > 0\} \tag{363}$$

There's a lot of neat things to say here, but I only have space to state the results:

- If C is an affine set and $x_0 \in C$, then

$$V = C - x_0 \triangleq \{x - x_0 \mid x \in C\} \tag{364}$$

  is a *subspace*, i.e. closed under sums and scalar multiplication.

[73] The **interior of a set** $C \subseteq \mathbb{R}^n$, denoted $\textbf{int}\, C$, is the set of all points interior to C. A point $x \in C$ is interior to C if $\exists \epsilon > 0$ for which *all points* in the set

$$\{y \in R^n \mid ||y - x||_2 \leq \epsilon\}$$

are also in C.

- The solution set of a system of linear equations, $C = \{x \mid Ax = b\}$, is an affine set. The subspace associated with C is the *nullspace* of A.

Convex sets

A set C is ***convex*** *if it contains all* ***convex combinations****,*

$$\sum_{i=1}^{k} \theta_i x_i, \quad \textit{where} \quad \sum_i \theta_i = 1, \ \textit{and}\ \theta_i \geq 0 \tag{365}$$

Related terminology:

- ***convex hull*** *a set C, denoted* **conv**C*, is the set of all convex combinations of points in C.*

Cones

A set C is called a ***cone*** *if $(\forall x \in C)(\theta \geq 0)$ we have $\theta x \in C$. A set C is called a* ***convex cone*** *if it contains all* ***conic combinations****,*

$$\sum_{i=1}^{k} \theta_i x_i, \quad \textit{where} \quad \theta_i \geq 0 \tag{366}$$

of points in C. Related terminology:

- ***conic hull*** *of a set C is the set of all conic combinations of points in C.*

Bayesian Data Analysis

Contents

Bayesian Data Analysis **November 22, 2017**

Probability and Inference (Ch. 1)

Table of Contents Local *Written by Brandon McKinzie*

The process of Bayesian Data Analysis can be divided into the following 3 steps:

1. Setting up a *full probability model*.
2. Conditioning on observed data. Calculating the *posterior distribution* over unobserved quantities, given observed data.
3. Evaluating the fit of the model and implications of the posterior.

Notation: In general, we let θ denote unobservable vector quantities or population *parameters* of interest, and y as collected data. This means our *posterior* takes the form $p(\theta \mid y)$, and our *likelihood* takes the form $p(y \mid \theta)$.

Means and Variances of Conditional Distributions.

$$\mathbb{E}[u] = \mathbb{E}_v[\mathbb{E}_u[u \mid v]] \tag{367}$$
$$\mathrm{Var}[u] = \mathbb{E}_v[\mathrm{Var}[u \mid v]] + \mathrm{Var}[\mathbb{E}_u[u \mid v]] \tag{368}$$

Proofs

$$\begin{aligned}
\mathbb{E}[u] &= \int \mathrm{d}v \Pr[v] \int \mathrm{d}u \Pr[u \mid v] && (369)\\
&= \mathbb{E}_v[\mathbb{E}_u[u \mid v]] && (370)\\
\mathrm{Var}[u] &= \mathbb{E}\left[u^2\right] - \mathbb{E}[u]^2 && (371)\\
&= \mathbb{E}_v\left[\mathbb{E}_u\left[u^2 \mid v\right]\right] - \mathbb{E}_v[\mathbb{E}_u[u \mid v]]^2 && (372)\\
&= \mathbb{E}_v\left[\mathbb{E}_u\left[u^2 \mid v\right] - \mathbb{E}_u[u \mid v]^2\right] + \mathbb{E}_v\left[\mathbb{E}_u[u \mid v]^2\right] - \mathbb{E}_v[\mathbb{E}_u[u \mid v]]^2 && (373)\\
&= \mathbb{E}_v[\mathrm{Var}[u \mid v]] + \mathrm{Var}[\mathbb{E}_u[u \mid v]] && (374)
\end{aligned}$$

Transformation of Variables.

$$\Pr_v[v] = |J| \Pr_u\left[f^{-1}(v)\right] \tag{375}$$

$J_{i,j} = \frac{\partial u}{\partial v} = \frac{\partial f^{-1}(v)}{\partial v}$

where u and v have the same dimensionality, and $|J|$ is the determinant of the Jacobian of the transformation $u = f^{-1}(v)$. When working with parameters defined on the open unit interval, $(0, 1)$, we often use the logistic transformation:

$$\mathrm{logit}(u) = \log\left(\frac{u}{1-u}\right) \tag{376}$$
$$\mathrm{logit}^{-1}(v) = \log\left(\frac{e^v}{1+e^v}\right) \tag{377}$$

Standard Probability Distributions[74].

Distribution	Notation	Density Function
Beta	$\text{Beta}(\alpha, \beta)$	$p(\theta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}$
Inverse-gamma	$\text{Inv-gamma}(\alpha, \beta)$	$p(\theta) = \frac{\beta^\alpha}{\Gamma(\alpha)}\theta^{-(\alpha+1)}e^{-\beta/\theta}$
Normal (univariate)	$\text{N}(\mu, \sigma^2)$	$p(\theta) = \frac{1}{\sqrt{2\pi}\sigma}\exp\big(-\frac{1}{2\sigma^2}(\theta-\mu)^2\big)$
Scaled inverse-χ^2	$\text{Inv-}\chi^2(\nu, s^2)$	$\theta \sim \text{Inv-gamma}(\frac{\nu}{2}, \frac{\nu}{2}s^2)$

[74]The **gamma function**, $\Gamma(x)$, is defined as

$$\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}\mathrm{d}t \tag{378}$$

or simply as $(x-1)!$ if $x \in \mathbb{Z}^+$.

Since we'll be referring to the binomial model frequently, below are the main distributions for reference:

$$p(y \mid \theta) = Bin(y \mid n, \theta) = \binom{n}{y} \theta^y (1-\theta)^{n-y} \tag{379}$$

$$p(\theta \mid y) \propto \theta^y (1-\theta)^{n-y} \tag{380}$$

$$\theta \mid y \sim \text{Beta}(y+1, n-y+1) \tag{381}$$

where y is the number of successes out of n trials. We assume a uniform prior over the interval [0, 1].

Posterior as compromise between data and prior information. Intuitively, the prior and posterior distributions over θ should have some general relationship showing how the process of observing data y updates our distribution on θ. We can use the identities from the previous chapter to see that

$$\mathbb{E}[\theta] = \mathbb{E}_y[\mathbb{E}_\theta[\theta \mid y]] \tag{382}$$

$$\text{Var}[\theta] = \mathbb{E}_y[\text{Var}[\theta \mid y]] + \text{Var}[\mathbb{E}_\theta[\theta \mid y]] \tag{383}$$

where:

→ Equation 382 states the obvious: our prior expectation for θ is the expectation, taken over the distribution of possible data, of the posterior expectation.

→ Equation 383 states: *the posterior variance is on average smaller than the prior variance*, by an amount that depends on the variation [in posterior means] over the distribution of possible data. Stated another way: we can reduce our uncertainty with regard to θ by larger amounts with models whose [expected] posteriors are strongly informed by the data.

Informative Priors. We now discuss some the issues that arise in assigning a prior distribution $p(\theta)$ that reflects substantive information. Instead of using a uniform prior for our binomial model, let's explore the prior $\theta \sim \text{Beta}(\alpha, \beta)$[75]. Now our posterior takes the form

$$\theta \mid y \propto \text{Beta}(\alpha + y, \beta + n - y) \tag{384}$$

The property that the posterior follows the same parametric form as the prior is called **conjugacy**.

[75] Assume we can select reasonable values for α and β.

Conjugacy, families, and sufficient statistics. Formally, if $\mathcal{F}$ is a class of sampling distributions $\{\Pr_i[y \mid \theta]\}$, and $\mathcal{P}$ is a class of prior distributions, $\{\Pr_j[\theta]\}$, then the class $\mathcal{P}$ is **conjugate** for $\mathcal{F}$ if[76]

$$\forall \Pr[\cdot \mid \theta] \in \mathcal{F}, \Pr[\cdot] \in \mathcal{P}: \quad \Pr[\theta \mid y] \in \mathcal{P} \tag{385}$$

Probability distributions that belong to an **exponential family** have natural conjugate prior distributions. The class $\mathcal{F}$ is an exponential family if all its members have the form,

$$\Pr[y_i \mid \theta] = f(y_i) g(\theta) e^{\phi(\theta)^T u(y_i)} \tag{386}$$

$$\Pr[y \mid \theta] = \left(\prod_{i=1}^{n} f(y_i) \right) g(\theta)^n \exp\left(\phi(\theta)^T \sum_{i=1}^{n} u(y_i) \right) \tag{387}$$

$$\propto g(\theta)^n \exp\left(\phi(\theta)^T t(y) \right) \tag{388}$$

where

- $y = (y_1, \ldots, y_n)$ denotes n iid observations.
- $\phi(\theta)$ is called the **natural parameter** of $\mathcal{F}$.
- $t(y) = \sum_{i=1}^{n} u(y_i)$ is said to be a **sufficient statistic** for θ, because the likelihood for θ depends on the data y only through the value of $t(y)$.

Normal distribution with known variance[77]. Consider a single scalar observation y drawn from $N(\theta, \sigma^2)$, where we assume σ^2 is known. The family of conjugate prior densities for the Gaussian likelihood as well as our choice of parameterization are, respectively,

Defining our conjugate prior

$$p(\theta) \propto e^{A\theta^2 + B\theta + C} \tag{390}$$

$$p(\theta) \propto e^{-\frac{1}{2\tau_0^2}(\theta - \mu_0)^2} \tag{391}$$

By definition, this implies that the posterior should also be normal. Indeed, after some basic arithmetic/substitutions, we find

Writing our posterior precision and mean.

$$\Pr[\theta \mid y] \propto \exp\left(-\frac{1}{2\tau_1^2}(\theta - \mu_1)^2 \right) \tag{392}$$

$$\mu_1 = \frac{\frac{1}{\tau_0^2}\mu_0 + \frac{1}{\sigma^2} y}{\frac{1}{\tau_0^2} + \frac{1}{\sigma^2}} \quad \text{and} \quad \frac{1}{\tau_1^2} = \frac{1}{\tau_0^2} + \frac{1}{\sigma^2} \tag{393}$$

[76]In English: A class of prior distributions is conjugate for a class of sampling distributions if, for any pair of sampling distribution and prior distribution [from those two respective classes], the associated posterior distribution is *also* in the same class of prior distributions.

[77]The following will be useful to remember:

$$\int_{-\infty}^{\infty} e^{-ax^2 + bx + c} \mathrm{d}x = \frac{\pi}{a} e^{\frac{b^2}{4a} + c} \tag{389}$$

where we see that the posterior **precision** (inverse of variance) equals the prior prior precision plus the data precision. We can see the posterior mean μ_1 expressed as a weighted average of the prior mean and the observed value[78] y, with weights proportional to the precisions.

Normal distribution with unknown variance. Now, we assume the mean θ is known, and the variance σ^2 is unknown. The likelihood for a vector $y = (y_1, \ldots, y_n)$ of n iid observations is

$$\Pr\left[y \mid \sigma^2\right] \propto (\sigma^2)^{-n/2} \exp\left(-\frac{n}{2\sigma^2} v\right) \tag{394}$$

$$v := \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta)^2 \tag{395}$$

Computing our likelihood for n IID observations.

where v is the sufficient statistic. The corresponding conjugate prior density is the inverse-gamma. This and our choice for parameterization (how we define α and β) are, respectively,

$$\Pr\left[\sigma^2\right] \propto (\sigma^2)^{-(\alpha+1)} e^{-\beta/\sigma^2} \tag{396}$$

$$\sigma^2 \sim \text{Inv-}\chi^2(\nu_0, \sigma_0^2) = \text{Inv-gamma}(\tfrac{\nu_0}{2}, \tfrac{\nu_0}{2}\sigma_0^2) \tag{397}$$

Defining our conjugate prior.

All that's left is computing our posterior,

$$p(\sigma^2 \mid y) \propto p(\sigma^2) p(y \mid \sigma^2) \tag{398}$$

$$\sigma^2 \mid y \sim \text{Inv-}\chi^2\left(\nu_0 + 2, \frac{\nu_0\sigma_0^2 + nv}{\nu_0 + n}\right) \tag{399}$$

Jeffrey's Invariance Principle. An approach for defining noninformative prior distributions. Let $\phi = h(\theta)$, where the function h is one-to-one. By transformation of variables,

$$p(\phi) = p(\theta)\left|\frac{d\theta}{d\phi}\right| = p(\theta)|h'(\theta)|^{-1} \tag{400}$$

Jeffrey's principle is to basically take the above equation as a true equivalence – there is no difference between finding $p(\theta)$ and applying the equation above [to get $p(\phi)$] and directly finding $p(\phi)$.

Let $J(\theta)$ denote the **Fisher information** for θ, defined as

$$J(\theta) = \mathbb{E}\left[\left(\frac{d \log \Pr\left[y \mid \theta\right]}{d\theta}\right)^2 \mid \theta\right] = -\mathbb{E}\left[\frac{d^2 \log \Pr\left[y \mid \theta\right]}{d\theta^2} \mid \theta\right] \tag{401}$$

Jeffrey's prior model defines the noninformative prior density as $\Pr\left[\theta\right] \propto [J(\theta)]^{1/2}$. We can work out that this model is indeed invariant to parameterization[79].

[78] For now, we're considering the single data point case.

[79] Evaluate $J(\phi)$ at $\theta = h^{-1}(\phi)$. You should find that $J(\phi)^{1/2} = J(\theta)^{1/2}|\frac{d\theta}{d\phi}|$

Asymptotics and Connections to Non-Bayesian Approaches (Ch. 4)

Normal Approximations to the Posterior Distribution. If the posterior $\Pr\left[\theta \mid y\right]$ is unimodal and roughly symmetric, it can be convenient to approximate it by a normal distribution. Here we'll consider a quadratic approximation via the Taylor series expansion up to second-order,

$$\log \Pr\left[\theta \mid y\right] = \log \Pr\left[\hat{\theta} \mid y\right] + \frac{1}{2}(\theta - \hat{\theta})^T \left[\frac{d^2}{d\theta^2} \log \Pr\left[\theta \mid y\right]\right]_{\theta=\hat{\theta}} (\theta - \hat{\theta}) \tag{402}$$

where $\hat{\theta}$ is the posterior mode. The remainder terms of higher order fade in importance relative to the quadratic term when θ is close to $\hat{\theta}$ and n is large. We'd like to cast this into a normal distribution. First, let

$$I(\theta) \triangleq -\frac{d^2}{d\theta^2} \log \Pr\left[\theta \mid y\right] \tag{403}$$

which we will refer to as the **observed information**. We can then rewrite our approximation as[80]

$$\Pr\left[\theta \mid y\right] \approx \mathcal{N}(\hat{\theta}, [I(\hat{\theta})]^{-1}) \tag{406}$$

Under the normal approximation, the posterior distribution is summarized by its mode, $\hat{\theta}$, and the curvature of the posterior density, $I(\hat{\theta})$; that is, asymptotically, these are sufficient statistics.

[80] I also found it helpful to explicitly write the substitution after raising eq 402 by power of e (all logs are assumed natural logs)

$$\begin{aligned}\Pr\left[\theta \mid y\right] &= e^{\log \Pr\left[\hat{\theta} \mid y\right] - \frac{1}{2}(\theta-\hat{\theta})^T\left[I(\hat{\theta})\right]^{-1}(\theta-\hat{\theta})} && (404)\\ &= \Pr\left[\hat{\theta} \mid y\right] e^{-\frac{1}{2}(\theta-\hat{\theta})^T\left[I(\hat{\theta})\right]^{-1}(\theta-\hat{\theta})} && (405)\end{aligned}$$

Example. Let $y_1, \ldots, y_n$ be independent observations from $\mathcal{N}(\mu, \sigma^2)$. Define $\theta := (\mu, \log\sigma)$ as the parameters of interest, and assume a uniform prior[81] $\Pr[\theta]$. Recall that (equation 3.2 in textbook)

$$\Pr[\theta = (\mu, \log\sigma) \mid y] \propto \sigma^{-(n+2)} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \mu)^2\right) \tag{408}$$

Note that $\sum_{i=1}^{n}(y_i - \bar{y}) = 0$

$$= \sigma^{-(n+2)} \exp\left(-\frac{1}{2\sigma^2}\left[n(\bar{y} - \mu)^2 + \sum_{i=1}^{n}(y_i - \bar{y})^2\right]\right) \tag{409}$$

$$= \sigma^{-(n+2)} \exp\left(-\frac{1}{2\sigma^2}\left[n(\bar{y} - \mu)^2 + (n-1)s^2\right]\right) \tag{410}$$

where $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2$ is the sample variance of the y_i's. The sufficient statistics are $\bar{y}$ and s^2. To construct the approximation, we need the second derivatives of the log posterior density[82],

$$\log \Pr[\mu, \log\sigma \mid y] = \text{const} - n\log\sigma - \frac{1}{2\sigma^2}\left(n(\bar{y} - \mu)^2 + (n-1)s^2\right) \tag{411}$$

in order to compute $I(\hat{\theta})$. After computing first derivatives, we find that the posterior mode is

$$\hat{\theta} = (\hat{\mu}, \log\hat{\sigma}) = \left(\bar{y}, \log\left(\sqrt{\frac{n-1}{n}}s\right)\right) \tag{412}$$

We then compute second derivatives and evaluate at $\theta = \hat{\theta}$ to obtain $I(\hat{\theta})$. Combining all this into the final result:

$$\Pr[\mu, \log\sigma \mid y] \approx \mathcal{N}(\hat{\theta}, [I(\hat{\theta})]^{-1}) \tag{413}$$

$$= \mathcal{N}\left(\begin{pmatrix}\bar{y} \\ \log\hat{\sigma}\end{pmatrix}, \begin{pmatrix}\hat{\sigma}^2/n & 0 \\ 0 & 1/(2n)\end{pmatrix}\right) \tag{414}$$

where $\hat{\sigma}^2/n$ is the variance along the μ dimension, and $1/(2n)$ is the variance along the $\log\sigma$ direction. This example was just meant to illustrate, with a simple case, how we work through constructing the approximate normal distribution.

[81] Recall from Ch 3.2 that the uniform prior on $\mu, \log\sigma$ is

$$\Pr\left[\mu, \sigma^2\right] \propto (\sigma^2)^{-1} \tag{407}$$

where we continue to assume μ is uniform in $[0, 1]$ for some reason.

[82] I'm not sure why $n + 2$ has seemingly turned into n.

Large-Sample Theory. Asymptotic normality of the posterior distribution: as more data arrives from the same underlying distribution $f(y)$, the posterior distribution of the *parameter vector* θ approaches multivariate normality, even if the true distribution of the data is not within the parametric family under consideration.

Suppose the data are modeled by a parametric family, $\Pr[y \mid \theta]$, with a prior distribution $\Pr[\theta]$. If the true distribution, $f(y)$, is included in the parametric family (i.e. if $\exists \theta_0 : f(y) = \Pr[y \mid \theta_0]$), then it's also true that **consistency** holds[83]: $\Pr[\theta \mid y]$ converges to a point mass at the true parameter value, θ_0 as $n \to \infty$.

[83]So, what if the true $f(y)$ is *not* included in the parametric family? In that case, there is no longer a true value θ_0, but its role in the theoretical result is replaced by a value θ_0 that makes the model distribution $\Pr[y \mid \theta]$ closest to the true distribution $f(y)$, in a technical sense involving the **Kullback-Leibler divergence**.

Gaussian Process Models (Ch. 21)

Written by Brandon McKinzie

Since I like to begin by motivating what we're going to talk about, and since BDA doesn't really do this, I'm going to start with an excerpt from chapter 15 of Kevin Murphy's book:

> *In supervised learning, we observe some inputs $\boldsymbol{x}_i$ and some outputs y_i. We assume that $y_i = f(\boldsymbol{x}_i)$, for some unknown function f, possibly corrupted by noise. The optimal approach is to infer a distribution over functions given the data, $p(f \mid \boldsymbol{X}, \boldsymbol{y})$, and then to use this to make predictions given new inputs, i.e., to compute*
>
> $$p(y_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \int p(y_* \mid f, \boldsymbol{x}_*) p(f \mid \boldsymbol{X}, \boldsymbol{y}) \mathrm{d}f \tag{415}$$
>
> ***Gaussian Processes*** *or GPs define a prior over functions $p(f)$ which can be converted into a posterior over functions $p(f \mid \boldsymbol{X}, \boldsymbol{y})$ once we've seen some data.*

Gaussian Process Regression (20.1). We write a GP as $\mu \sim \mathrm{GP}(m, k)$ (μ is now taking the place of f from Kevin Murphy's notation), parameterized in terms of a mean function m and a covariance function k. Remember, μ is supposed to represent the predictor function for obtaining output predictions given inputs, $y = \mu(x)$. Instead of making the usual assumption that there is some "best" function μ^* and trying to learn it via fitting a parameterized $\hat{\mu}(\boldsymbol{\theta})$, we are going *full meta*[84] (i.e. full Bayesian) and learning the distribution over predictors.

Apparently, we only need to consider a finite (and arbitrary) set of points $x_1, \ldots, x_n$ to consider when evaluating any given μ. The **GP prior** on μ is defined as

$$\mu(x_1), \ldots, \mu(x_n) \sim \mathcal{N}\left(\{m(x_1), \ldots, m(x_n)\}, K(x_1, \ldots, x_n)\right) \tag{416}$$

with mean m and covariance K[85]. The covariance function k specifies the covariance between the process at any two points, with K an $n \times n$ covariance matrix with $K_{p,q} = k(x_p, x_q)$. The covariance function controls the smoothness of realizations from the GP[86] and the degree of shrinkage towards the mean.

[84] *What if there are like, a whole space of different predictors, man? Like, what if there is an infinite sea of predictor functions, all with their own unique traits and quirks? Woah.*

[85] Don't confuse the notation – $\mathcal{N}$ uses covariance K as an argument, while *GP* uses covariance *function* k as an argument.

[86] In English: How similar we expect different samples of μ to look as a function of x. The reason this was weird to think about at first is because I'm used to thinking about covariance/smoothness over x rather than sampled *functions* of x. Meta.

A common choice the squared exponential,

$$k(x, x') = \tau^2 \exp\left(-\frac{|x - x'|^2}{2\ell^2}\right) \tag{417}$$

where τ controls the magnitude and ℓ the smoothness of the function.

Elements of Statistical Learning

Contents

ESL

Linear Regression

Table of Contents Local *Written by Brandon McKinzie*

- Assumption: The **regression function** $\mathbb{E}[Y|X]$ is linear[87] in the inputs $X_1, \ldots, X_p$.
- Perform well for...
  - Small numbers of training cases.
  - Low signal/noise.
  - Sparse data.

6.1.1 Models and Least-Squares

- The **model**:

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j \tag{3.1}$$

- Most popular **estimation method** is least-squares.

$$\begin{aligned} RSS(\beta) &= \sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2 && (3.2) \\ &= (\boldsymbol{y} - \boldsymbol{X}\beta)^T (\boldsymbol{y} - \boldsymbol{X}\beta)^T && (3.3) \end{aligned}$$

  which is reasonable if training observations (x_i, y_i) represent independent random draws from their population[88].

- First two derivatives wrt to parameter vector β:

$$\begin{aligned} \frac{\partial RSS}{\partial \beta} &= -2\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\beta) \\ \frac{\partial^2 RSS}{\partial \beta \partial \beta^T} &= 2\boldsymbol{X}^T\boldsymbol{X} \end{aligned} \tag{3.4}$$

- Assuming that $\boldsymbol{X}$ has full column rank so that $\boldsymbol{X}^T\boldsymbol{X}$ is positive definite[89], set first derive to 0 to obtain the unique solution:

[87] or reasonably approximated as linear

[88] and/or if y_i's conditionally indep given the x_i's.

[89] A matrix is positive definite if it's symmetric and all its eigenvalues are positive. **What would we do here if X were *not* full column rank?**. **Answer:** $\boldsymbol{X}$ columns may not be linearly independent if, e.g., two inputs were perfectly correlated $\boldsymbol{x}_2 = 3\boldsymbol{x}_1$. The fitted $\hat{\mathbf{y}}$ will still be projection onto $C(\boldsymbol{X})$, but there will be more than 1 way (not unique) to express that projection. Occurs most often when one or more (qualitative) inputs are coded in a redundant fashion.

$$\hat{\beta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{3.6}$$

- **Geometry of Least Squares** I GET IT NOW!
  - The $(p+1)$ column vectors of $\boldsymbol{X}$ span a subspace of $\mathbb{R}^N$.[90].
  - Minimizing $RSS(\beta) = ||\boldsymbol{y} - \boldsymbol{X}\beta||^2$ is choosing $\hat{\beta}$ such that the **residual vector** $\boldsymbol{y} - \hat{\boldsymbol{y}}$ is orthogonal to this subspace[91]. Stated another way, (the optimal) $\hat{y}$ is the *orthogonal projection of* $\boldsymbol{y}$ *onto the column space of* X.
  - Since $\hat{\mathbf{y}} = \boldsymbol{X}\hat{\beta}$, we can define this projection matrix (aka hat matrix), denoted as $\boldsymbol{H}$, where

$$\begin{aligned}\hat{\mathbf{y}} &= \boldsymbol{X}\hat{\beta} \\ &= \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \\ &= \boldsymbol{H}\boldsymbol{y}\end{aligned} \tag{3.7}$$

Why $Var(\hat{\beta}) = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\sigma^2$**.**

- Note: **Variance-covariance matrix** $\equiv$ Covariance matrix.
- Can express the **correlation matrix** in terms of the covariance matrix:

$$corr(\boldsymbol{X}) = \big(diag(\Sigma)\big)^{-1/2}\Sigma\big(diag(\Sigma)\big)^{-1/2} \tag{418}$$

  or, equivalently, the correlation matrix can be seen as the covariance matrix of the standardized random variables $X_i/\sigma(X_i)$.

- Recall from decision theory that, when we want find a function $f(X)$ for predicting some $Y \in \mathbb{R}$, we can do this by *minimizing the risk* (aka the prediction error EPE(f)). This is accomplished first by defining a loss function. Here we will use the squared error loss $L(Y, f(X)) = (Y - f(X))^2$. We can express $EPE(f)$ as an integral over all values that Y and X may take on (i.e. the joint distribution). Therefore, we can factor the joint distribution and define $f(x)$ via minimizing EPE piecewise (meaning at each value of $X = x$.) This whole description is written mathematically below.

$$EPE(f) = \mathbb{E}\left[Y - f(X)\right]^2 \tag{2.9}$$
$$= \int \left[y - f(x)\right]^2 f_{XY}(x,y)dxdy \tag{2.10}$$
$$= \mathbb{E}_X\Big[\mathbb{E}_{Y|X}\Big[(Y - f(X))^2|X\Big]\Big] \tag{2.11}$$

[90]This is the **column space** $C(\boldsymbol{X})$ of $\boldsymbol{X}$. It is the space of $\boldsymbol{X}v$ $\forall v \in \mathbb{R}^N$, since the produce Xv is just a linear combination of the columns in X with coefficients v_i.

[91]Interpret: $X\beta$ will always lie *somewhere* in this subspace, but we want β such that, when we subtract each component (WOAH JUST CLICKED) from the prediction, they cancel exactly,i.e. $y_i - (\boldsymbol{X}\beta)_i = 0$ for all dimensions i in $C(X)$. The resultant vector $\boldsymbol{y} - \hat{y}$ will only contain components outside this subspace, hence it is orthogonal to it by definition.

and therefore, the best predictor of Y is a function $f : \mathbb{R}^p \to \mathbb{R}$ that satisfies, for each x value separately

$$f(x) = \arg\min_c \mathbb{E}_{Y|X}\left[(Y-c)^2|X\right] \tag{2.12}$$
$$= \mathbb{E}\left[Y|X=x\right] \tag{2.13}$$

which essentially defines what is meant by $\mathbb{E}\left[Y|X=x\right]$, also referred to as the **conditional mean**[92].

Bias-Variance Tradeoff

The test MSE, for a given value x_0, can always be decomposed into the sum of three fundamental quantities:

$$\mathbb{E}\left[y_0 - \hat{f}(x_0)\right]^2 = Var(\hat{f}(x_0)) + \left[Bias(\hat{f}(x_0))\right]^2 + Var(\epsilon) \tag{419}$$

which is interpreted as the *test MSE*: the average test MSE that we would obtain if we repeatedly estimated f using a large number of training sets, and tested each at x_0. The **overall test MSE** can be computing the average (of this average) over all possible values of x_0 in the TEST set.

- What **bias** means here: On the other hand, bias refers to the error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model. For example, linear regression assumes that there is a linear relationship between Y and $X_1, \ldots, X_p$. It is unlikely that any real-life problem truly has such a simple linear relationship, and so performing linear regression will undoubtedly result in some bias in the estimate of f. In Figure 2.11, the true f is substantially non-linear, so no matter how many training observations we are given, it will not be possible to produce an accurate estimate using linear regression. In other words, linear regression results in high bias in this example. However, in Figure 2.10 the true f is very close to linear, and so given enough data, it should be possible for linear regression to produce an accurate estimate. Generally, more flexible methods result in less bias.

- Returning now to the case where know (aka assume) that the true relationship between X and Y is linear

$$Y = X^T\beta + \epsilon \tag{2.26}$$

and so *in this particular case* the least squares estimates are unbiased.

- This is the proof: (relies on the fact that $Var(\beta) = 0$ since β is the true (NON RANDOM)

[92] At the same time, don't forget that least-squared error assumption was built-in to this derivation.

vector we are estimating)[93]

$$
\begin{aligned}
Var[\hat{\beta}] &= Var\Big[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}\Big] && (420)\\
&= Var\Big[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T(\boldsymbol{X}\beta+\epsilon)\Big] && (421)\\
&= Var\Big[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{X}\beta+(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\epsilon\Big] && (422)\\
&= Var\Big[\beta+(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\epsilon\Big] && (423)\\
&= Var\Big[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\epsilon\Big] && (424)\\
&= \mathbb{E}\Big[\Big((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\epsilon\Big)\Big((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\epsilon\Big)^T\Big] && (425)\\
&= \Big((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\Big)\ \mathbb{E}\Big[\epsilon\epsilon^T\Big]\ \Big(\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\Big) && (426)\\
&= \Big((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\Big)\ \sigma^2\ \Big(\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\Big) && (427)\\
&= \sigma^2\Big((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\Big)\Big(\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\Big) && (428)\\
&= \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1} && (429)
\end{aligned}
$$

where we have assumed that the X are FIXED (not random)[94] and so the variance of (some product of Xs) $\times$ ϵ is like taking the variance with a constant out front. We've also assumed that X^TX (and thus its inverse too) is symmetric, apparently.

6.1.2 Subset Selection (3.3)

- Two reasons why we might not be satisfied with 3.6:
  1. Prediction accuracy. Often have low bias, high variance. May improve if shrink coefficients. Sacrifices some bias to reduce variance.
  2. Interpretation. Sacrifices some of the small details.
- Appears that subset selection refers to retaining a subset of the *predictors* $\hat{\beta}_i$ and discarding the rest.
- Doing this can often exhibit high variance, even if lower prediction error.

6.1.3 Shrinkage Methods (3.4)

- Shrinkage methods are considered *continuous* (as opposed to subset selection) and don't suffer as much from high variability.

[93] Also, $\forall a \in \mathbb{R} : Var(a+X) = Var(X)$

[94] another way of stating this is that we took the variance given (or conditioned on) each X

Linear Methods for Classification (Ch. 4)

Logistic Regression (4.4). Motivation: we want to model $\Pr[G = k \mid X = x]$, for each of our K classes, via <u>linear functions in x</u>.

$$\log \frac{\Pr[G = k \mid X = x]}{\Pr[G = K \mid X = x]} = (\beta_0)_k + \beta_k^T x \qquad \forall k \in [1, K-1] \tag{430}$$

$$\Pr[G = k \mid X = x] = \frac{e^{\beta_{0k} + \beta_k^T x}}{1 + \sum_{\ell=1}^{K-1} e^{\beta_{0\ell} + \beta_\ell^T x}} \quad \forall k \in [1, K-1] \tag{431}$$

$$\triangleq p_k(x; \theta) \tag{432}$$

where $\theta \triangleq \{\beta_{01}, \beta_1^T, \ldots, \beta_{(K-1)0}, \beta_{K-1}^T\}$.

ESL

Naive Bayes

Table of Contents Local *Written by Brandon McKinzie*

Appropriate when dimension p of feature space is large. It assume that given a class $G = j$, the features X_k are independent:

$$f_j(X) \equiv f_j((X_1, X_2, \dots, X_p)^T) = \prod_{k=1}^{p} f_{jk}(X_k) \tag{433}$$

which can simplify estimation [of the class-conditional probability densities $f_j(X)$] dramatically: The individual class-conditional marginal densities f_{jk} can each be estimated *separately* using 1D kernel density estimates.

6.4.1 Boosting Methods (10.1)

Terminology:

- **Weak Classifier**: one whose error rate is only slightly better than random guessing.

The AdaBoost algorithm.

1. Initialize observation weights $w_i := 1/N$, $i = 1, 2, \ldots, N$.
2. For $m = 1$ to M[95]:
   (a) Fit classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i} \tag{434}$$
   (c) Compute $\alpha_m = \log\left((1 - \mathrm{err}_m)/\mathrm{err}_m\right)$.
   (d) Update $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$, $i = 1, \ldots, N$.
3. Output $G(x) = \sum_{i=1}^{m} \alpha_m G_m(x)$.

[95] where M is the number of weak classifiers (trees) that we want to train.

Basis Expansions and Regularization (Ch 5)

Introduction. Core idea is to augment/replace the vector of inputs X with additional variables, which are transformations of X, and then use linear models in this new space of derived inputs features. Specifically, we model a *linear basis expansion* in X:

$$f(X) = \sum_{m=1}^{M} \beta_m h_m(X) \tag{435}$$

$h_m(X) : \mathbb{R}^p \rightarrowtail \mathbb{R}$.

where $h_m(X)$ is the mth transformation of X, $m = 1, \ldots, M$.

Piecewise Polynomials and Splines. Assume that X is one-dimensional. We will explore increasingly complex cases that build off each other.

- **Piecewise-constant**. Take, for example, the case where we have 3 basis functions:

$$h_1(X) = I(X < \xi_1), \quad h_2(X) = I(\xi_1 \le X < \xi_2), \quad h_3(x) = I(\xi_2 \le X) \tag{436}$$

Solving for β_i in each derivative of $RSS(\beta)$ w.r.t. β_i, for $f(X) = \sum_{i=1}^{3} \beta_i h_i(X)$, yields $\hat{\beta}_i = \bar{Y}_i$, the mean of Y in the ith (of 3) regions. Note that this is the degree-0 case: piecewise constant.

- **Piecewise-linear**. Instead of just learning the best-fit constant β_i for the ith region, we can now also learn the best *slope* for the line in that region. This means we have 3 additional parameters to fit, and $f(X)$ becomes:

$$\begin{aligned} f(X) &= \beta_1 I_1 + \beta_2 I_2 + \beta_3 I_3 + \beta_4 I_1 \cdot X + \beta_5 I_2 \cdot X + \beta_6 I_3 \cdot X && (437) \\ &= (\beta_1 + \beta_4 X) I_1 + (\beta_2 + \beta_5 X) I_2 + (\beta_3 + \beta_6 X) I_3 && (438) \end{aligned}$$

where I've denoted the ith identity function from 436 as I_i.

- **Continuous piecewise-linear**. We would typically prefer the lines to meet (have the same value) at the region boundaries of $X = \xi_1$ and $X = \xi_2$. In other words, require that:

$$\begin{aligned} f(\xi_1^-) &= \beta_1 + \beta_4 \xi_1 = \beta_2 + \beta_5 \xi_1 = f(\xi_1^+) && (439) \\ f(\xi_2^-) &= \beta_2 + \beta_5 \xi_2 = \beta_3 + \beta_6 \xi_2 = f(\xi_2 +) && (440) \end{aligned}$$

which also reduces our free parameters from 6 to 4[96]. We can express this constraint

[96]Because, for example, now $\beta_1 = \beta_2 + (\beta_5 - \beta_4)\xi_1$.

more directly by using a basis that incorporates them[97].

$$h_1(X) = 1, \quad h_2(X) = X, \quad h_3(X) = (X - \xi_1)_+, \quad h_4(X) = (X - \xi_2)_+ \tag{441}$$

In general, **an order-M spline with knots $\xi_j,\ j = 1, \ldots, K$ is a piecewise-polynomial of order M, and has continuous derivatives up to order $M - 2$.**

B-splines. Let ξ_0 and ξ_K denote two **boundary knots**, which typically define the domain over which we wish to evaluate our spline. Define the augmented knot sequence τ such that

$$\tau_i \le \tau_2 \le \cdots \le \tau_M \le \xi_0 \tag{442}$$
$$\tau_{M+j} = \xi_j,\ j = 1, \cdots, K \tag{443}$$
$$\xi_{K+1} \le \tau_{K+M+1} \le \tau_{K+M+2} \le \cdots \le \tau_{K+2M} \tag{444}$$

It is customary to make all $\tau_{i,i \le M} = \xi_0$ and $\tau_{j,j \ge K+M+1} = \xi_{K+1}$.

The ith B-spline basis function of order m ($m \le M$) for the knot sequence τ is denoted by $B_{i,m}(x)$, and defined recursively as follows:

$$B_{i,1}(x) = \begin{cases} 1 & \tau_i \le x \le \tau_{i+1} \\ 0 & \text{otherwise} \end{cases} \quad i = 1, \ldots, K + 2M - 1 \tag{445}$$

$$B_{i,m}(x) = \frac{x - \tau_i}{\tau_{i+m-1} - \tau_i} B_{i,m-1}(x) + \frac{\tau_{i+m} - x}{\tau_{i+m} - \tau_{i+1}} B_{i+1,m-1}(x) \quad i = 1, \ldots, K + 2M - m \tag{446}$$

[97]Note: I was not able to derive this form from the previous equations and constraints. Moving on because not important right now.

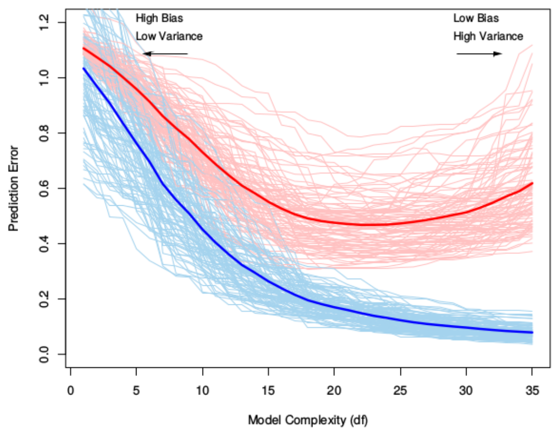
Bias, Variance, and Model Complexity. We first define the three main quantities[98]:

Generalization (Test) Error $$\mathrm{Err}_{\mathcal{T}} = \mathbb{E}_{(X,Y)\sim p}\left[L(Y, \hat{f}(X)) \mid \mathcal{T}\right] \tag{447}$$

Expected Prediction Error $$\mathrm{Err} = \mathbb{E}\left[L(Y, \hat{f}(X))\right] = \mathbb{E}_{\mathcal{T}}\left[\mathrm{Err}_{\mathcal{T}}\right] \tag{448}$$

Training Error $$\overline{\mathrm{err}} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i)) \tag{449}$$

Below we see **$\mathrm{Err}_{\mathcal{T}}$** and **$\overline{\mathrm{err}}$** as a function of model complexity. The thick solid lines represent their averages (i.e. approximate the expectations).



[98] Y is target variable, X is vector of inputs, $\mathcal{T}$ is training set. We assume that X and Y can be sampled from some true/underlying/unknown joint distribution $p(X, Y)$.

Bias-Variance Decomposition. Let $Y = f(X) + \varepsilon$, $\mathbb{E}[\epsilon] = 0$, $\text{Var}[\varepsilon] = \sigma_\varepsilon^2$. We derive the expected prediction error of a fit $\hat{f}(X)$ at an input point $X = x_0$, using squared-error loss:

$$\begin{aligned}
\text{Err}(x_0) &= \mathbb{E}_{\varepsilon,\hat{f}}\left[(Y - \hat{f}(x_0))^2 \mid X = x_0\right] && (450)\\
&= \mathbb{E}_{\varepsilon,\hat{f}}\left[(f - \hat{f}(x_0) + \varepsilon)^2\right] && (451)\\
&= \mathbb{E}_{\varepsilon}\left[\varepsilon^2\right] + \mathbb{E}_{\hat{f}}\left[(f - \hat{f}(x_0))^2\right] && (452)\\
&= \sigma_\varepsilon^2 + \mathbb{E}_{\hat{f}}\left[\hat{f}(x_0)^2\right] + f^2 - 2f\mathbb{E}_{\hat{f}}\left[\hat{f}(x_0)\right] && (453)\\
&= \sigma_\varepsilon^2 + \mathbb{E}_{\hat{f}}\left[\hat{f}(x_0)^2\right] - \mathbb{E}_{\hat{f}}\left[\hat{f}(x_0)\right]^2 + \mathbb{E}_{\hat{f}}\left[\hat{f}(x_0)\right]^2 + f^2 - 2f\mathbb{E}_{\hat{f}}\left[\hat{f}(x_0)\right] && (454)\\
&= \sigma_\varepsilon^2 + \text{Var}\left[\hat{f}(x_0)\right] + \left(\mathbb{E}_{\hat{f}}\left[\hat{f}(x_0)\right]^2 - 2f\mathbb{E}_{\hat{f}}\left[\hat{f}(x_0)\right] + f^2\right) && (455)\\
&= \sigma_\varepsilon^2 + \text{Var}\left[\hat{f}(x_0)\right] + \left(\mathbb{E}_{\hat{f}}\left[\hat{f}(x_0)\right] - f\right)^2 && (456)\\
&= \sigma_\varepsilon^2 + \text{Var}\left[\hat{f}(x_0)\right] + \text{Bias}^2\left(\hat{f}(x_0)\right) && (457)
\end{aligned}$$

ε and $\hat{f}$ are independent.

Remember, $\text{Err}(x_0)$ is the MSE of the prediction at $X = x_0$, taken over all possible training sets (resulting in a trained model $\hat{f}$) (and the irreducible error from ε). The bias is the difference between the average estimate, $\mathbb{E}_{\hat{f}}\left[\hat{f}(x_0)\right]$, and the true mean of Y, $f(x_0)$. Finally, the variance measures how much the distribution over all possible $\hat{f}$ deviates from their average, $\mathbb{E}_{\hat{f}}\left[\hat{f}(x_0)\right]$.

BV Tradeoff for KNN

For simplicity, assume the set of inputs $\{x_i\}$=X is fixed, and the randomness arises from the associated y_i. As before, our modeling assumption is that $Y = f(X) + \epsilon$, with $\epsilon \sim \mathcal{N}\left(0, \sigma_\epsilon^2\right)$. It follows that $\mathbb{E}[Y] = f(X)$. Denote the KNN estimator $\hat{f}_k(x)$.

$$\begin{aligned}
\text{Err}_{KNN}(x_0) &= \mathbb{E}_{y\sim p(y|x_0)}\left[(y - \hat{f}_k(x_0))^2\right] && (458)\\
&= \sigma_\epsilon^2 + \left[f(x_0) - \frac{1}{k}\sum_{\ell=1}^{k} f(x_{(\ell)})\right]^2 + \frac{\sigma_\epsilon^2}{k} && (459)
\end{aligned}$$

Perhaps counter-intuitively, the complexity of $\hat{f}_k$ is $\propto \frac{1}{k}$ – the number of "effective parameters" for a KNN model over N data points is N/k. Increasing k will typically increase bias but decrease variance.

Bootstrap Methods. A general tool for assessing statistical accuracy. Seeks to estimate the conditional error $\text{Err}_{\mathcal{T}}$ but typically estimates well only the expected prediction error Err. Denote our training set $\mathcal{T}$ by $\boldsymbol{Z} = (z_1, \ldots, z_N)$, where $z_i = (x_i, y_i)$. The basic procedure is as follows:

1. We randomly draw datasets with replacement from $\boldsymbol{Z}$, with each sampled dataset, $\boldsymbol{Z}^{*b}$, having the same size, N, as $\boldsymbol{Z}$.
    - This is done B times, producing B bootstrap datasets.
2. Refit the model to each of the bootstrap datasets, and examine the behavior of the fits over the B replications.

Modern Mathematical Statistics with Applications

Contents

Modern Mathematical Statistics with Applications **February 08, 2020**

Probability (Ch. 2)

Table of Contents Local *Written by Brandon McKinzie*

Devore and Berk (2018). Probability. *Modern Mathematical Statistics with Applications.*

Exercise 31 [correct]

(a) ${}^5P_2 = 5!/3! = 20$

(b) ${}^5P_3 = 60$

(c) $\binom{5}{2} = 5!/(3!2!) = 10$

Exercise 35

iPod playlist has 100 songs, of which 10 are by the Beatles. Say you have shuffle on. What is the probability that the first Beatles song heard is the fifth song played?

It's just the probability that you don't hear a Beatles song 4x in a row, then do hear one:

$$(90/100)^4(10/100) = \frac{9^4}{10^4}\frac{1}{10} = 9^4 \times 10^{-5} = 0.0656$$

[wrong] You forgot to handle the fact that this is done without replacement, you dummy:

$$\frac{90}{100}\frac{89}{99}\frac{88}{98}\frac{87}{97}\frac{10}{96} = 0.0679$$

Gaussian Processes for Machine Learning

Contents

Gaussian Processes for Machine Learning **July 29, 2018**

Regression (Ch. 2)

Table of Contents Local *Written by Brandon McKinzie*

Rasmussen and Williams (2006). Regression. *Gaussian Processes for Machine Learning.*

Weight-space view (2.1). We review the standard probabilistic view of linear regression.

$X \in \mathbb{R}^{d \times n}$

$$f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{w} \tag{460}$$

$$y = f(\boldsymbol{x}) + \varepsilon \quad \text{where} \quad \varepsilon \sim \mathcal{N}(0, \sigma_n^2) \tag{461}$$

$$\text{[likelihood]} \qquad \boldsymbol{y} \mid X, \boldsymbol{w} \sim \mathcal{N}(X^T \boldsymbol{w}, \sigma_n^2 I) \tag{462}$$

$$\text{[prior]} \qquad \boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \Sigma_p) \tag{463}$$

$$\text{[posterior]} \qquad p(\boldsymbol{w} \mid \boldsymbol{y}, X) = \frac{p(\boldsymbol{y} \mid \boldsymbol{w}, X) p(\boldsymbol{w})}{p(\boldsymbol{y} \mid X)} \tag{464}$$

$$\sim \mathcal{N}(\frac{1}{\sigma_n^2} A^{-1} X \boldsymbol{y}, A^{-1}) \tag{465}$$

where $A = \sigma_n^{-2} X X^T + \Sigma_p^{-1}$, and we often set $\Sigma_p = I$. When analyzing the contour plots for the likelihood, remember that it is *not* a probability distribution, but rather it's interpreted as a function of $\boldsymbol{w}$, i.e. likelihood($\boldsymbol{w}$) := $\mathcal{N}(\boldsymbol{y}; X^T \boldsymbol{w}, \sigma_n^2 I)$.

Note that we often use Bayesian techniques without realizing it. For example, what does the following remind you of?

$$\ln p(\boldsymbol{w}) \propto \frac{1}{2} \boldsymbol{w}^T \Sigma_p \boldsymbol{w} \tag{466}$$

It's the l2 penalty from ridge regression (where typically $\Sigma_p = I$). We can also project the inputs to a higher-dimensional space, often referred to as the *feature space*, by passing them through feature functions $\phi(x)$. As we'll see later (ch 5), GPs actually tell us how to define the basis functions $h_i(x)$ which define the value of $\phi_i(x)$, the ith element of the feature vector. The author then proceeds to give an overview of the kernel trick.

Function-space view (2.2). Instead of inference over parameters $\boldsymbol{w}$, we can equivalently consider inference in function space with a **Gaussian process** (GP), formally defined as

> *A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.*

A GP is completely specified by its mean function and covariance function. Define mean function $m(\boldsymbol{x})$ and covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$ of a real [Gaussian] process $f(\boldsymbol{x})$ as

$$m(\boldsymbol{x}) = \mathbb{E}_f \left[f(\boldsymbol{x}) \right] \tag{467}$$

$$k(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}_f \left[(f(\boldsymbol{x}) - m(\boldsymbol{x})) \left(f(\boldsymbol{x}') - m(\boldsymbol{x}') \right) \right] \tag{468}$$

$$f(\boldsymbol{x}) \sim \mathcal{GP} \left(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}') \right) \tag{469}$$

Note that the expectations are over the *random variable* $f(\boldsymbol{x})$ for any *given* (non-random) $\boldsymbol{x}$. In other words, the expectation is over the space of possible functions, each evaluated at point $\boldsymbol{x}$. Concretely, this is often an expectation over the parameters $\boldsymbol{w}$, as is true for our linear regression example. We can now write our Bayesian linear regression model (with feature functions) as a GP.

$$\mathbb{E}_{\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \Sigma_p)} \left[f(\boldsymbol{x}; \boldsymbol{w}) \right] = \phi(\boldsymbol{x})^T \mathbb{E} \left[\boldsymbol{w} \right] = 0 \tag{470}$$

$$\mathbb{E}_{\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \Sigma_p)} \left[f(\boldsymbol{x}) f(\boldsymbol{x}') \right] = \phi(\boldsymbol{x})^T \mathbb{E} \left[\boldsymbol{w}\boldsymbol{w}^T \right] \phi(\boldsymbol{x}') = \phi(\boldsymbol{x})^T \Sigma_p \phi(\boldsymbol{x}') \tag{471}$$

A common covariance function is the **squared exponential** (a.k.a. the RBF kernel),

$$k(\boldsymbol{x}_p, \boldsymbol{x}_q) = \text{Cov} \left[f(\boldsymbol{x}_p), \; f(\boldsymbol{x}_q) \right] = \exp(-\frac{1}{2} |\boldsymbol{x}_p - \boldsymbol{x}_q|^2) \tag{472}$$

where it's important to recognize that, whereas we've usually seen this is the RBF kernel for the purposes of kernel methods on inputs, we are now using it specify the *covariance of the **outputs***.

Ok, so how do we sample some functions and plot them? Below is an overview for our current linear regression running example.

1. Choose a number of input points X_*. For our linear regression example, we could set this to `np.arange(-5, 5.1, 0.1)` to get evenly spaced x in $[-5, 5]$ in intervals of 0.1.
2. Write out the covariance matrix defined by $K_{p,q} = k(x_p, x_q)$ using our squared exponential covariance function, for all pairs of inputs.
3. We can now generate samples of function f, represented as a random vector with size equal to the number of inputs $|X_*|$, by sampling from the **GP prior**

$$\boldsymbol{f}_* \sim \mathcal{N}(\boldsymbol{0}, K(X_*, X_*)) \tag{473}$$

So far, we've only dealt with the GP *prior*. What do we do when we get labeled training observations? How do we make predictions on unlabeled test data? Well, for the simple case where our observations are noise free[99], that is we know $\{(\boldsymbol{x}_i, f_i) \mid i = 1, \ldots, n\}$, the joint

[99]For example, noise-free linear regression would mean we model y=f(x), implicitly defining $\varepsilon = 0$.

GP prior over the train set inputs X and test set inputs X_* is defined exactly how we did it earlier (zero mean, elementwise evaluation of k). In other words, our GP prior models the train outputs $\boldsymbol{f}$ and test outputs $\boldsymbol{f}_*$ as random vectors sampled via

$$\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right) \tag{474}$$

You may be wondering: *why are we talking about sampling from the prior on inputs? We already know the outputs!*, and you'd be correct. The way we obtain our posterior is by restricting our joint prior to only those functions that agree with the observed training data $X, \boldsymbol{f}$, which we can do by simply conditioning on them. Our posterior for sampling test outputs given test inputs X_* is thus

$$\begin{aligned} \boldsymbol{f}_* \mid X_*, X, \boldsymbol{f} \sim \mathcal{N}\Big(& K(X_*,X)K(X,X)^{-1}\boldsymbol{f}, \\ & K(X_*,X_*) - K(X_*,X)K(X,X)^{-1}K(X,X_*)\Big) \end{aligned} \tag{475}$$