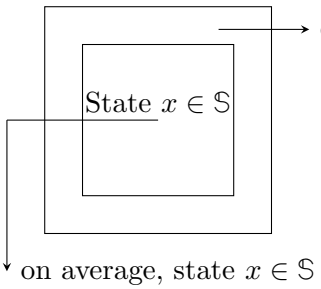
I Gibbs ensembles and a few of their consequences

A Maximum Entropy Principle

1. The strange thought experiment of J.W. Gibbs (1839-1903)
2. Recap
3. Maximum Entropy Principle (MEP)
4. Examples

1 The strange thought experiment of J.W. Gibbs (1839-1903)

Imagine a physical system, $\mathcal{E}$ (e.g. a gas), sitting in an "infinite heat bath"



∞ heat bath, temperature T
($\Leftrightarrow$ average (translational) kinetic energy of particles fixed)

on average, state $x \in \mathbb{S}$ is also at temperature T

(i) Discretize state: $x \in \mathbb{S} \leftrightarrow\ x \in \{1, 2, \cdots, s\}$ (later $\rightarrow$ continuum limit)

(ii) x represents the state of every particle, e.g. position (x, y, z) and velocity components (v_x, v_y, v_z) for each of $\sim 10^{27}$ gas molecules

(iii) s is very large ($s >> 1$)

(iv) $\mathcal{E}(x)$ total kinetic energy of system when it is in state x

Gibbs asks: What is the probability, p_x, of finding the system in state x, $\forall x \in \{1, 2, \cdots, s\}$?

But how could we possibly even have a clue? Surely, we need to know more than just the temperature T?

Nevertheless, Gibbs argues that

(a) On average, the gas will be at temperature T (in equilibrium with the heat bath):

$$\sum_{x=1}^{s} p_x \mathcal{E}(x) = \theta \tag{1}$$

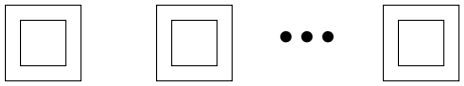
for some constant θ that depends on temperature.

(Or, if you prefer,

$$\int_x f(x)\mathcal{E}(x)dx = \theta$$

where f is a "probability density function" (pdf) instead of a "probability mass function" (pmf), and θ depends on T.)

(b) Now consider a (very) large ensemble of identical systems and heat baths



$n >> s >> 1$

State: X_1 X_2 X_n

in which case

$$p_x \approx \hat{p}_x \triangleq \frac{1}{n}\sum_{k=1}^{n} \mathbb{1}_{X_k=x} = \#\{k : X_k = x\}/n \tag{2}$$

($\hat{p}_x$ is known at the "empirical probability distribution", and this particular ensemble is known as the "canonical ensemble")

The claim is that, overwhelmingly, the vast majority of assignments of states $\{1, 2, \ldots, s\}$ to systems $1, 2, \cdots, n$ that are consistent with (1) and (2), i.e. consistent with

$$\sum_x \hat{p}_x \mathcal{E}(x) \approx \theta \tag{3}$$

yield a single empirical distribution $\hat{p}_x$ on $\{1, 2, \cdots, s\}$. (What's more, and a bit remarkable, is that this particular distribution is apparently the right distribution!)

Let's try to compute this magical distribution:

Overall, there are s^n ways to assign a states to each of the n systems, and through (2), any such assignment defines a particular empirical distribution

$$\hat{p} = (\hat{p}_1, \hat{p}_2, \cdots, \hat{p}_s)$$

For each empirical distribution $\hat{p}$, let $C(\hat{p})$ be the number of assignments that end up with the same $\hat{p}$.

Given $\hat{p}$, the number of systems in the state x is $n_x = triangleq \hat{p}_x n$. We can think of $C(\hat{p})$ as the number of ways to form s committees from n people, where there are $\hat{p}_1 n$ people in committee 1, $\cdots$, $\hat{p}_s n$ people in committee s, which leads us to the multinomial generalization of the binomial formula:

$$C(\hat{p}) = \frac{n!}{n_1! n_2! \cdots n_s!} \tag{4}$$

The numerator, $n!$, is the number of different ways to line up the n people. As for the denominator:

$n_1!$ is the number of different ways to line up the first n_1 people (each of which gives the same makeup of the first committee)

$n_2!$ is the number of different ways to line up the next n_2 people

$\vdots$

$n_s!$ is the number of different ways to line up the last n_s people

In order to study $C(\hat{p})$, Stirling's remarkably accurate approximation is just the right tool:

for $k \geq 1$	$e^{\frac{1}{12k+1}}$	$\leq k!/(k^k e^{-k}\sqrt{2\pi k})$	$\leq e^{\frac{1}{12k}}$
k	$\downarrow$		$\downarrow$
1	$1.0800\cdots$		$1.0869\cdots$
10	$1.0083\cdots$		$1.0089\cdots$
100	$1.0008\cdots$		$1.0008\cdots$

We will write $a_n = O(b_n)$ if a_n and b_n are two sequences with

$$|a_n| \leq B|b_n| \text{ for some } B \text{ and all } n$$

We want to maximize $C(\hat{p})$ over all $\hat{p}$ such that $\sum_x \hat{p}_x \mathcal{E}(x) \approx \theta$. Start by using Stirling to get a more useful expression for $C(\hat{p})$: go back to expression (4) for $C(\hat{p})$ and write $\mathcal{S}(C(\hat{p}))$ to mean (4), but with all factorials replaced by Stirling's approximation:

$$k! \to k^k e^{-k}\sqrt{2\pi k}$$

Then, at least heuristically (i.e. non-rigorously—rigor will come later):

$$\begin{aligned}
\frac{1}{n}\log C(\hat{p}) &= \frac{1}{n}\log\frac{C(\hat{p})}{\mathcal{S}(C(\hat{p}))} + \frac{1}{n}\log \mathcal{S}(C(\hat{p})) \quad \text{(replace factorials by their Stirling approximations)} \\
&= O(\frac{1}{n^2}) + \frac{1}{n}\log\frac{n^n e^{-n}\sqrt{2\pi n}}{\prod_{i=1}^s (n\hat{p}_x)^{\hat{p}_x n} e^{-\hat{p}_x n}\sqrt{2\pi \hat{p}_x n}} \quad \text{(see HW problem for the first term)} \\
&= O(\frac{1}{n^2}) + \frac{1}{n}\log\frac{n^n e^{-n}\sqrt{2\pi n}}{n^n e^{-n}\prod_{i=1}^s \hat{p}_x^{\hat{p}_x n}\sqrt{2\pi \hat{p}_x n}} \text{ (use } \sum_x \hat{p}_x = 1) \\
&= O(\frac{1}{n^2}) + O(\frac{\log n}{n}) - \sum_{x=1}^s \hat{p}_x \log \hat{p}_x \\
&= -\sum_{x=1}^s \hat{p}_x \log \hat{p}_x + O(\frac{\log n}{n}) \\
&\Rightarrow \frac{1}{n}\log C(\hat{p}) \to H(\hat{p}) \triangleq -\sum_{x=1}^s \hat{p}_x \log \hat{p}_x \leftarrow \text{Shannon's Entropy! (more on this later)}
\end{aligned}$$

So $C(\hat{p}) \approx e^{nH(\hat{p})}$ when n large.

Recall that $C(\hat{p})$ = # assignments of states to systems that yield the particular empirical distribution $\hat{p} = (\hat{p}_1, \cdots, \hat{p}_s)$. The claim was that, given the constraint in (3), $C(\hat{p})$ is very peaked around its maximum. Let us compute its maximum (again, non-rigorously):

First, bring in the constraints:

(i) $\sum_{x=1}^s \hat{p}_x \mathcal{E}(x) = \theta$ because the temperature is fixed. (Actually, this is approximate for $n < \infty$, but close enough.)

(ii) $\sum_{x=1}^s \hat{p}_x = 1$

Let

$$\Omega = \left\{ q : \sum_{x=1}^{s} q_x \mathcal{E}(x) = \theta, \sum_{x=1}^{s} q_x = 1 \right\}$$

The goal is to compute

$$\hat{p}^* = \underset{q \in \Omega}{\operatorname{argmax}}\, C(q) = \underset{q \in \Omega}{\operatorname{argmax}}\, H(q)$$

and this calls for Lagrange (1736-1813) multipliers: first choose q to maximize

$$H(q) + \lambda \sum_{x=1}^{s} q_x \mathcal{E}(x) + \gamma \sum_{x=1}^{s} q_x \tag{5}$$

and then choose λ and γ to satisfy the two constraints, (i) and (ii). Fix $\tilde{x} \in \{1, 2, \cdots, s\}$:

$$\begin{aligned} 0 &= \frac{\partial}{\partial q_{\tilde{x}}} [- \sum_{x=1}^{s} q_x \log q_x + \lambda \sum_{x=1}^{s} q_x \mathcal{E}(x) + \gamma \sum_{x=1}^{s} q_x] \\ &= -\log q_{\tilde{x}} - 1 + \lambda \mathcal{E}(\tilde{x}) + \gamma \quad \forall\, \tilde{x} \\ &\Longrightarrow q_x = e^{-1} e^{\gamma} e^{\lambda \mathcal{E}(x)} = \frac{1}{Z_\lambda} e^{\lambda \mathcal{E}(x)} \quad \forall x \in 1, \ldots, s \end{aligned}$$

where λ is determined by (i) and then Z_λ, the normalizing constant, is chosen to satisfy (ii).

Remarks

1. In terms of temperature T in Kelvin and energy $\mathcal{E}$ in Joules:

$$q_x = \frac{1}{z_T} e^{-\frac{1}{KT} \mathcal{E}(x)}$$

   where K is "Boltzmann's constant"

2. In the continuum limit (discretization $\to 0 \leftrightarrow s \to \infty$), this becomes a density function $q_x \to q(x)$

2 Recap

The argument, in brief, went like this:

(i) We seek a probability distribution p on $\{1, 2, \cdots, s\}$ that satisfies

$$\sum_{x=1}^{s} p_x \mathcal{E}(x) = \theta$$

where $\mathcal{E}$ is some (given) function on $\{1, 2, \cdots, s\}$ and θ is some given constant.

(ii) In general, there will be many solutions. If we imagine n selections from $\{1, 2, \cdots, s\}$ (say $X_1, X_2, \cdots, X_n$) and let

$$n_x = \#\{k : X_k = x\}/n \quad \text{for } x = 1, \cdots, s$$

then it might make sense to use

$$\hat{p}_x \triangleq \frac{n_x}{n} \text{ (the "empirical distribution")}$$

provided that

(a) $\sum_{x=1}^{s} \hat{p}_x \mathcal{E}(x) \approx \theta$

(b) $\binom{n}{n_1\ n_2\ \cdots\ n_s}$ is as large as possible, given (a)

(iii) Since

$$\binom{n}{n_1\ n_2\ \cdots\ n_s} \approx e^{nH(\hat{p})} \text{ for large } n$$

we look for

$$p = \underset{q \in \Omega}{\operatorname{argmax}}\ e^{nH(q)} \tag{6}$$

where $\Omega = \{q : q \text{ pmf on } \{1, 2, \cdots, s\} \text{ and } \sum_{x=1}^{s} q_x \mathcal{E}(x) = \theta\}$.

(iv) The solution to (6) is

$$p(x) = \frac{1}{Z_\lambda} e^{\lambda \mathcal{E}(x)}$$

where $Z_\lambda = \sum_{x=1}^{s} e^{\lambda \mathcal{E}(x)}$ and λ is chosen so that p satisfies

$$\sum_{x=1}^{s} p_x \mathcal{E}(x) = \theta$$

3 Maximum Entropy Principle (MEP)

The MEP is just the generalization when there are $c \geq 1$ constraints (instead of just one).

Suppose we know that some probability, p, on $\{1, 2, \cdots, s\}$ satisfies

$$\left.\begin{array}{l} \sum_{x=1}^{s} p_x \mathcal{E}_1(x) = \theta_1 \\ \vdots \\ \sum_{x=1}^{s} p_x \mathcal{E}_c(x) = \theta_c \end{array}\right\} c \text{ linear constraints on } p = (p_1, \cdots, p_s)$$

(Notice that these could also be written as $E_p[\mathcal{E}_k(X)] = \theta_k, k = 1, \cdots, c$)

Let

$$\Omega = \{q : q \text{ pmf satisfying the constraints}\}$$

MEP: Choose

$$p = \underset{q \in \Omega}{\operatorname{argmax}}\ H(q)$$

The solution is derived exactly as in the special case $c = 1$, just completed, leading to

$$p_x = \frac{1}{Z_{\vec{\lambda}}} e^{\sum_{k=1}^{c} \lambda_k \mathcal{E}_k(x)}$$

where $\vec{\lambda} = (\lambda_1, \ldots, \lambda_c)$ is chosen to satisfy the c constraints, and Z, which is a function of $\vec{\lambda}$, is the normalizing constant.

In physics:

p	"Gibbs distribution" (aka "Boltzmann distribution")
$Z_{\lambda_1,\lambda_2,\cdots,\lambda_c}$	"partition function" (a function of $\vec{\lambda}$ whose derivatives and discontinuities have physical interpretations, e.g. the temperatures at which phase transitions occur)

In statistics:

$p = p(x : \vec{\lambda})$	"exponential family" indexed by $\vec{\lambda}$
$\mathcal{E}_1(x), \cdots, \mathcal{E}_c(x)$	"sufficient statistics" (sufficient for estimating $\vec{\lambda}$ - as we shall see)

4 Examples

1. (MEP, discrete setting)

   Find the maximum entropy distribution p on $\{1, 2, 3\}$ satisfying

   $$E_p[X^2] = 2$$

   Here, $\mathcal{E}(x) = x^2 \Rightarrow p_x = \frac{1}{Z_\lambda} e^{\lambda x^2}$

   $$\Rightarrow p_1 = \frac{1}{Z_\lambda} e^{\lambda}, \;\; p_2 = \frac{1}{Z_\lambda} e^{4\lambda}, \;\; p_3 = \frac{1}{Z_\lambda} e^{9\lambda}$$
   $$\Rightarrow Z_\lambda = e^{\lambda} + e^{4\lambda} + e^{9\lambda}$$
   $$\Rightarrow 2 = E_p[X^2] = \frac{1 \cdot e^{\lambda}}{e^{\lambda} + e^{4\lambda} + e^{9\lambda}} + \frac{4 \cdot e^{4\lambda}}{e^{\lambda} + e^{4\lambda} + e^{9\lambda}} + \frac{9 \cdot e^{9\lambda}}{e^{\lambda} + e^{4\lambda} + e^{9\lambda}}$$
   $$\Rightarrow e^{\lambda} - 2e^{4\lambda} - 7e^{9\lambda} = 0 \qquad \text{Ugh!}$$

   (Soon, we will devise an elegant numerical approach)

2. (MEP in the continuum)

   Given a probability density function $p(x)$, the entropy (now sometimes called the differential entropy) is defined as

   $$H(p) = -\int_{-\infty}^{\infty} p(x) \log p(x) dx$$

   Find the maximum entropy density subject to

   $$E_p[X^2] = 1, \;\; E_p[X] = 0.$$

   Our derivation still goes through (though not quite trivially) and we get

   $$p(x) = \frac{1}{Z_{\lambda_1,\lambda_2}} e^{\lambda_1 x^2 + \lambda_2 x}$$

This is enough to guess the exact solution, satisfying the two constraints:

$$p(x) \sim N(0,1)$$

i.e. $\lambda_2 = 0$, $\lambda_1 = -\frac{1}{2}$, and $p(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$

I Gibbs ensembles and a few of their consequences

B Large Deviation Principle

1. Setup
2. Distribution of $\hat{p}(X_{1:n})$
3. Properties of $D(q\|h)$
4. Conditioning on a large deviation
5. Sanov's Theorem
6. Linear constraints and the exponential families
7. The partition function and the computation of $\vec{\lambda}$

1 Setup

(i) h pmf on $\{1, 2, \cdots, s\}$

(ii) $\mathcal{E} : \{1, 2, \cdots, s\} \to \mathbb{R}$

(iii) $X_{1:n} \sim$ iid h

(iv) Observe $\frac{1}{n}\sum_{k=1}^{n} \mathcal{E}(X_k) = \theta$

This observation is a large deviation if θ is far from

$$\mathbb{E}_h[\mathcal{E}(X)]$$

which would be unexpected, because $\frac{1}{n}\sum_{k=1}^{n} \mathcal{E}(x_k) \underset{LLN}{\longrightarrow} \mathbb{E}_h[\mathcal{E}(X)]$.
LDP is the study of

$$\hat{p} = \hat{p}(X_{1:n})$$

the empirical distribution that produced the large deviation:

$$\mathbb{E}_{\hat{p}}[\mathcal{E}(X)] = \frac{1}{n}\sum_{k=1}^{n} \mathcal{E}(X_k) = \theta \tag{1}$$

One way to think about (1) is to note that $\hat{p}$ is the distribution that puts a mass of $\frac{1}{n}$ at each value of X_k, and hence

$$\hat{p}_x = \frac{\#\{k : X_k = x\}}{n} = \frac{1}{n}\sum_{k=1}^{n} \mathbb{1}_{X_k = x}$$

$$\Rightarrow \quad \mathbb{E}_{\hat{p}}[\mathcal{E}(X)] = \sum_{x=1}^{s} \mathcal{E}(x)\hat{p}(x) = \sum_{x=1}^{s} \frac{1}{n}\sum_{k=1}^{n} \mathcal{E}(x)\mathbb{1}_{X_k = x}$$

$$= \frac{1}{n}\sum_{k=1}^{n}\sum_{x=1}^{s} \mathcal{E}(x)\mathbb{1}_{X_k = x} = \frac{1}{n}\sum_{k=1}^{n} \mathcal{E}(X_k)$$

2 Distribution of $\hat{p}(X_{1:n})$

Given n, let $q = (q_1, \ldots, q_s)$ be an empirical distribution on $\{1, \ldots, s\}$. What is the probability that $\hat{p}$, which depends on $X_{1:n}$ ($\hat{p} = \hat{p}(X_1, \ldots, X_n)$) is equal to q? For each $x \in \{1, \ldots, s\}$, let $n_x = nq_x$. Then:

$$\begin{aligned}
\mathbb{P}\left(\hat{p} = q\right) &= \binom{n}{n_1 n_2 \cdots n_s} \prod_{x=1}^{s} h_x^{n_x} \\
&\approx e^{nH(q)} \prod_{x=1}^{s} h_x^{n_x} \quad \text{(as shown previously, see I.A)} \\
&= e^{nH(q)} e^{\sum_{x=1}^{s} n_x \log h_x}
\end{aligned}$$

Hence

$$\begin{aligned}
\frac{1}{n} \log \mathbb{P}\left(\hat{p} = q\right) &\approx H(q) + \sum_{x=1}^{s} \frac{n_x}{n} \log h_x \\
&= -\left(\sum_{x=1}^{s} q_x \log q_x - \sum_{x=1}^{s} q_x \log h_x \right) \\
&= -\sum_{x=1}^{s} q_x \log \frac{q_x}{h_x} = -D(q\|h)
\end{aligned}$$

where

$$D(q\|h) \triangleq \sum_{x=1}^{s} q_x \log \frac{q_x}{h_x}$$

is the "Kullback-Leibler divergence" aka "Kullback-Leibler distance" (but it's not symmetric and hence not a distance) aka "relative entropy."

3 Properties of $D(q\|h)$

(i) $D(q\|h) \geq 0 \quad \forall\, q, h$ pmfs

(ii) $D(q\|h) = 0 \Leftrightarrow q = h$

Both of these properties arise from Jensen's (1859-1925) inequality:

> *If $G(x)$ is a convex function, then for any random variable X,*
>
> $$\mathbb{E}[G(X)] \geq G(\mathbb{E}[X])$$
>
> *What is more, equality holds if and only if G is constant on the support of X.*

To connect to the properties of $D(q\|h)$, define $G(x) = -\log(x)$, which is convex, and then apply Jensen's inequality to the random variable

$$Y = \frac{h_X}{q_X}$$

where $X \sim q$:

$$\begin{aligned} D(q\|h) &= \sum_{x=1}^{s} q_x \log\left(\frac{q_x}{h_x}\right) = \mathbb{E}_q\left[\log\left(\frac{q_X}{h_X}\right)\right] \\ &= \mathbb{E}_q\left[-\log\left(\frac{h_X}{q_X}\right)\right] \geq -\log\left(\mathbb{E}_q\left[\frac{h_X}{q_X}\right]\right) \quad \text{(Jensen's inequality)} \\ &= -\log\left(\sum_{x=1}^{s} q_x \frac{h_x}{q_x}\right) = -\log\left(\sum_{x=1}^{s} h_x\right) = -\log(1) = 0 \end{aligned}$$

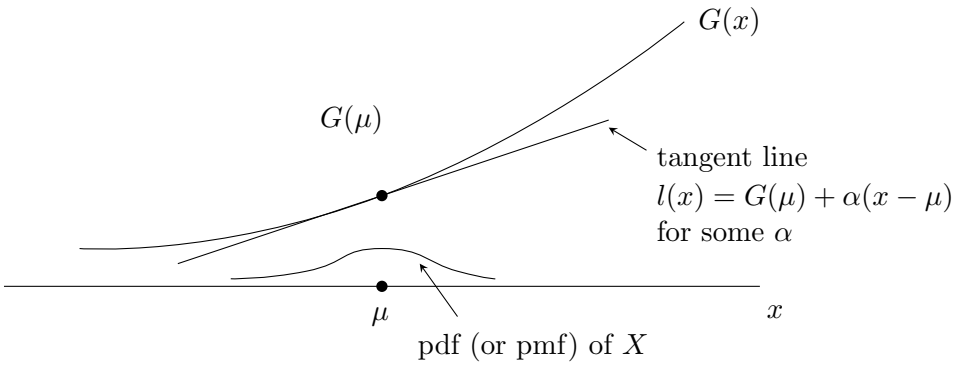
Aside on Jensen's Inequality. You will prove Jensen's inequality in the homework, using induction and the properties of convex functions. But let's make a picture that is far more revealing: Given a random variable X with expected value $\mu = \mathbb{E}[X]$, and a convex function $G(x)$, Jensen's inequality asserts that

$$\mathbb{E}[G(X)] \geq G(\mathbb{E}[X]) = G(\mu)$$

Now consider the straight line defined by

$$l(x) = G(\mu) + \alpha(x - \mu)$$

At $x = \mu$, $l(x) = G(x)$, so the line and the function G share at least one point. Now adjust α so that the line l is tangent to the curve G. Here is the picture and the reasoning behind the inequality:



Now, using the figure, we can reason that

$$E[G(X)] \geq E[l(X)] = G[\mu] + \alpha\, \mathbb{E}[X - \mu] = G(\mu)$$

since $\mathbb{E}[X] = \mu$.

(Note: This "proof" is not general because it assumes the existence of a tangent line. But without too much fuss, it can be extended to a proper proof. Alternatively, the inequality can be proven by induction—see HW.)

4 Conditioning on a large deviation

Returning to the large deviation discussion, and sticking (for now) with an informal derivation, we have

$$\mathbb{P}(\hat{p}(X_{1:n}) = q) \approx e^{-nD(q\|h)}$$

and to find the most likely $\hat{p}$, given that

$$\hat{p} \in \Omega \quad \text{where} \quad \Omega \triangleq \{q : q \text{ is a pmf and } \sum_x q_x \mathcal{E}(x) \approx \theta\}$$

we will compute

$$p \triangleq \underset{q \in \Omega}{\text{argmax}}(-D(q\|h)) \quad (= \underset{q \in \Omega}{\text{argmin}}(D(q\|h)))$$

This calls, again, for Lagrange multipliers:

For all $\tilde{x} = 1, 2, \cdots, s$

$$\begin{aligned} 0 &= \frac{\partial}{\partial q_{\tilde{x}}}(-\sum_x q_x \log q_x + \sum_x q_x \log h_x + \lambda \sum_x q_x \mathcal{E}(x) + \gamma \sum_x q_x) \\ &= -\log q_{\tilde{x}} - 1 + \log h_{\tilde{x}} + \lambda \mathcal{E}(\tilde{x}) + \gamma \\ &\Rightarrow q_{\tilde{x}} = e^{-1} e^{\gamma} h_{\tilde{x}} e^{\lambda \mathcal{E}(\tilde{x})} \end{aligned}$$

giving us

$$p_x = \frac{1}{Z_\lambda} h_x e^{\lambda \mathcal{E}(x)}, \quad x = 1, \cdots, s$$

If instead there were multiple constraints,

$$\hat{p} \in \Omega \quad \text{where} \quad \Omega \triangleq \{q : q \text{ is a pmf and } \sum_x q_x \mathcal{E}_k(x) \approx \theta_k, k = 1, \ldots, c\}$$

then by the same reasoning

$$p_x = \frac{1}{Z_{\vec{\lambda}}} h_x e^{\sum_{k=1}^c \lambda_k \mathcal{E}_k(x)}, \quad x = 1, \cdots, s$$

Of course, whether there is just one constraint ($c = 1$) or many, we still need to figure out how to compute $\vec{\lambda}$ from the equations

$$\mathbb{E}_p[\mathcal{E}_k(X)] \approx \theta_k \ \forall k = 1 : c$$

We will come back to this shortly, but first we will state the large deviation principle in a much more general and precise setting.

5 Sanov's (1919-1968) Theorem

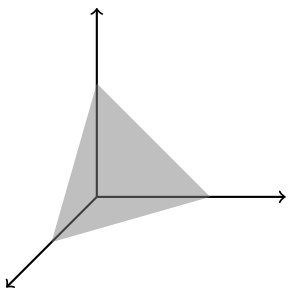
Here's the setup:

$$X_{1:n} \sim \text{ iid } h \text{ and } \hat{p}_x = \frac{1}{n} \sum_{i=1}^n \mathbb{1}_{X_i = x}$$

and let

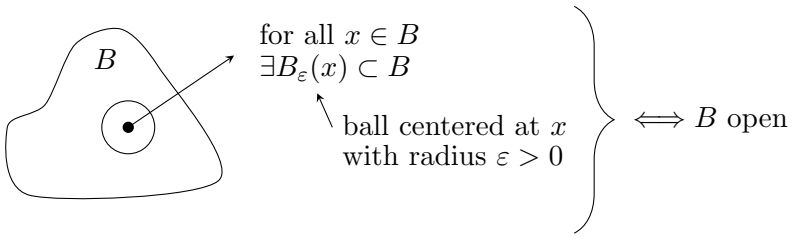
$$\$ = \{q : q \text{ pmf on } \{1, 2, \cdots, s\}\} \quad \text{(which is sometimes called the "probability simplex")}$$

e.g. $s = 3$



and let B be an open subset of $\mathbb{S}$

$B \subset \mathbb{S}$



and define $\|\alpha - \beta\| = \max_x |\alpha_x - \beta_x|$ for any $\alpha, \beta \in \mathbb{S}$.

(a) What is $\mathbb{P}(\hat{p} \in B)$ when n is large?

(b) Given that $\hat{p} \in B$, what can we say about $\hat{p}$?

Theorem (LDP)

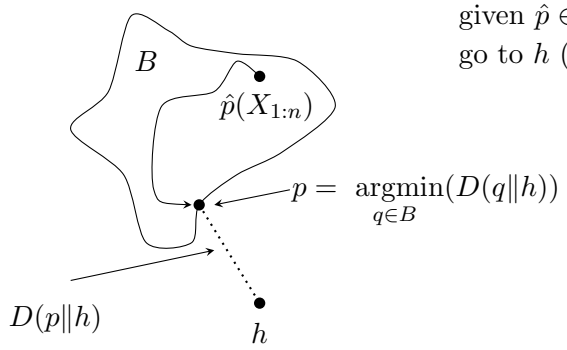
a $\lim_{n\to\infty} \frac{1}{n} \log \mathbb{P}(\hat{p}(X_{1:n}) \in B) = -\inf_{q \in B} D(q\|h)$

b if $p \triangleq \operatorname{argmin}_{q \in B} D(q\|h)$ is unique, then

$$\lim_{n\to\infty} \mathbb{P}(\|\hat{p}(X_{1:n}) - p\| > \varepsilon \,\big|\, \hat{p}(X_{1:n}) \in B) = 0 \qquad \forall \varepsilon > 0$$

Remarks

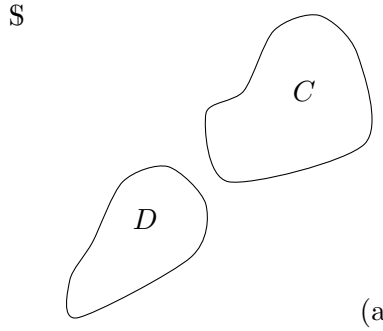
1. The notion of distance doesn't matter. We could have used $\|\alpha - \beta\| = \sum_x |\alpha_x - \beta_x|$ instead of $\max_x |\alpha_x - \beta_x|$, for example.
2. Here's a good summary, using a picture:



given $\hat{p} \in B$, it tries to
go to h (LLN), but can only get to p

3. The "large deviation" is that $\hat{p} \in B$

4. Why insist on B open? Consider



$B = C \cup D \cup E$
Suppose $\underset{q \in B}{\operatorname{argmin}} D(q\|h) = E$
Since E is irrational,
and since $\hat{p}$ is rational ($\hat{p}_x = \#\{k : X_k = x\}/n$),
$\hat{p}$ can never equal E, even given that $\hat{p} \in B$

(a single isolated point with irrational coordinates)

Technicality: demanding that B is open is too strong. It would be enough to require that $B \subset \overline{B^o}$ (B is in the closure of the interior of B).

5. If $h \in B$, then $p = h$ and $D(p\|h) = 0$ and

$$\mathbb{P}(\hat{p}(X_{1:n}) \in B) \to 1 \text{ as } n \to \infty$$

6. I drew $\hat{p}$ on the boundary of B (when $h \notin B$). Is it?

Fix h and consider $D(q\|h)$ as a function of q on $

Claim it's convex: $D(q\|h) = \sum_x q_x \log \frac{q_x}{h_x}$

$$\Rightarrow \frac{\partial D(q\|h)}{\partial q_i} = \log q_i + 1 - \log h_i$$

$$\Rightarrow \frac{\partial^2}{\partial q_i q_j} D(q\|h) = \delta_{ij}/q_i \Rightarrow H = \begin{pmatrix} 1/q_1 & & & \\ & 1/q_2 & & 0 \\ & & \ddots & \\ & 0 & & 1/q_s \end{pmatrix} \text{ (Hessian Matrix)}$$

$$\Rightarrow v^T H v > 0 \ \ \forall \ \|v\| = 1$$

But $v^T H v$ is the second derivative of $D(q\|h)$ in direction v

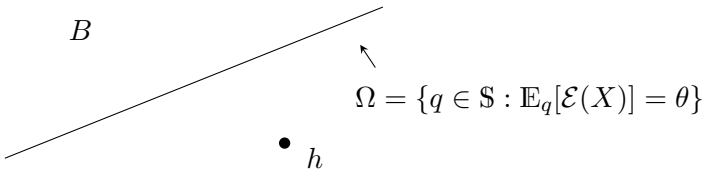
$\Rightarrow D(q\|h)$ has a global minimum (namely, $q = h$) and no local minima.

Hence $\underset{q\in B}{\operatorname{argmin}} D(q\|h)$ cannot occur in the interior of B (since that would be a local minimum).

Remark: but $\underset{q\in B}{\operatorname{argmin}} D(q\|h)$ is not necessarily unique, though B convex would be a sufficient condition for uniqueness.

6 Linear constraints and the exponential families

Here we will consider the important special case in which B is a half-plane (or more generally an intersection of multiple half-planes):



where Ω is the boundary of B. (Keep in mind that since $\mathbb{E}_q[\mathcal{E}(X)] = \sum_{x=1:s} q_x \mathcal{E}(x)$, the constraint imposed by Ω, namely $\mathbb{E}_q[\mathcal{E}(X)] = \theta$, is *linear* in $q_1, \ldots, q_s$, which are the variables of interest.)

According to the LDP, if

$$p \triangleq \underset{q\in\Omega}{\operatorname{argmin}} D(q\|h) \tag{2}$$

then for every $\epsilon > 0$

$$\mathbb{P}(\|\hat{p}(X_{1:n}) - p\| > \varepsilon \big| \hat{p}(X_{1:n}) \in B) \to 0$$

and for large n

$$\mathbb{P}(\hat{p} \in B) \approx e^{-nD(p\|h)}$$

We have already computed the form of the solution to (2):

$$p_x = p_x(\lambda) = \frac{1}{Z_\lambda} h(x) e^{\lambda \mathcal{E}(x)}$$

More generally, if Ω includes multiple linear constraints

$$\Omega = \{q \in \mathbb{S} : \mathbb{E}_q[\mathcal{E}_k(X)] = \theta_k,\ k = 1, \ldots, c\}$$

then

$$p_x = p_x(\vec{\lambda}) = \frac{1}{Z_{\vec{\lambda}}} h(x) e^{\vec{\lambda}\cdot\mathcal{E}(x)} \tag{3}$$

where $\mathcal{E} = (\mathcal{E}_1, \ldots, \mathcal{E}_c)$, $\vec{\lambda} = (\lambda_1, \ldots, \lambda_c)$, and $\vec{\lambda} \cdot \mathcal{E} = \sum_{k=1}^{c} \lambda_k \mathcal{E}_k(x)$. (Often it is more convenient to write $p(x; \vec{\lambda})$ instead of $p_x(\vec{\lambda})$. Going forward, we will use both.)

Remarks

1. The distributions defined in (3) are the *exponential families*. There is one family for every h_x and every set of *sufficient statistics*, $\mathcal{E}_1, \cdots, \mathcal{E}_c$. Examples of exponential families are the binomial, Poisson, and multinomial (all discrete), and the normal, gamma, exponential, and Dirichlet (all continuous). There are other parameterizations; the one in (3) is sometimes called the canonical parametrization.

2. The prototypical exponential family for continuous random variables consists of the normal densities, $\mathcal{N}(\mu, \sigma^2)$:

$$f(x) = f(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2} \quad x \in \{-\infty, \infty\}$$

If $\lambda_1 = -1/2\sigma^2$ and $\lambda_2 = \mu/\sigma^2$ then f can be written in the form

$$f(x; \vec{\lambda}) = \frac{1}{Z_\lambda} e^{\lambda_1 x^2 + \lambda_2 x} = \frac{e^{\lambda_1 x^2 + \lambda_2 x}}{\int_{-\infty}^{\infty} e^{\lambda_1 y^2 + \lambda_2 y} dy}$$

where h is 1, a constant.

3. In general, h does not have to be a pmf or a pdf, but it cannot be negative.

4. The construction in (3) is not unique. Constants can be moved between λ_k and $\mathcal{E}_k(x)$ and between $h(x)$ and $Z_{\vec{\lambda}}$.

5. Suppose that we start with an exponential family (3) as a model: h and $\mathcal{E}_1, \cdots, \mathcal{E}_c$ are assumed known, and our goal is to estimate the unknown parameters $\lambda_1, \ldots, \lambda_c$ from data $X_{1:n}$, which is assumed to have been drawn iid from the model. One way to do this is by the procedure known as maximum-likelihood estimation, which we will discuss later in §II. The important point for now is that the **joint** distribution for $X_{1:n}$ is again an exponential family, this time on the random vector $\vec{X} = (X_1, \ldots, X_n)$:

$$\begin{aligned} p(x_1, \ldots, x_n; \vec{\lambda}) &= \prod_{i=1:n} p(x_i; \vec{\lambda}) \\ &= \prod_{i=1:n} \frac{1}{Z_{\vec{\lambda}}} h(x_i) e^{\vec{\lambda} \cdot \mathcal{E}(x_i)} \\ &= \frac{1}{Z_{\vec{\lambda}}^n} \left(\prod_{i=1:n} h(x_i) \right) e^{\vec{\lambda} \cdot \sum_{i=1:n} \mathcal{E}(x_i)} \\ &= \frac{1}{\tilde{Z}_{\vec{\lambda}}} \tilde{h}(\vec{x}) e^{\vec{\lambda} \cdot \tilde{\mathcal{E}}(\vec{x})} \end{aligned}$$

where

$$\tilde{Z}_{\vec{\lambda}} = Z_{\vec{\lambda}}^n \quad \tilde{h}(\vec{x}) = \prod_{i=1:n} h(x_i) \quad \tilde{\mathcal{E}}(\vec{x}) = \sum_{i=1:n} \mathcal{E}(x_i)$$

Notice that the new sufficient statistics $\tilde{\mathcal{E}}_k$, $k = 1, \ldots, c$, are just n times the empirical averages of the old sufficient statistics $\mathcal{E}_k$, $k = 1, \ldots, c$.

7 The partition function and the computation of $\vec{\lambda}$

Finally, we turn our attention to the problem of solving for the Lagrange multipliers, $\lambda_1, \ldots, \lambda_c$, in the special cases involving linear constraints.

For convenience, we will write λ for $\vec{\lambda}$ and E_λ for $E_{p(\vec{\lambda})}$. We wish to solve the system of equations

$$E_\lambda[\mathcal{E}_k(X)] = \theta_k, \quad k = 1 : c$$

where $h, \mathcal{E}_1, \ldots, \mathcal{E}_c$ and $\theta_1, \ldots, \theta_c$ are given, and

$$p_x = p_x(\lambda) = \frac{1}{Z_\lambda} h(x) e^{\sum_1^c \lambda_k \mathcal{E}_k(x)}$$

The key to computing λ is the "partition function," as is evident from the following proposition:

Proposition

1. $\dfrac{\partial}{\partial \lambda_k} \log Z_\lambda = E_\lambda[\mathcal{E}_k(X)] \quad \forall\, k$

2. $\dfrac{\partial^2}{\partial \lambda_k \partial \lambda_l} \log Z_\lambda = \mathbb{C}_\lambda[\mathcal{E}_k(X), \mathcal{E}_l(X)] \quad \forall\, k, l$ (i.e. the covariance of $\mathcal{E}_k(X)$ & $\mathcal{E}_l(X)$)

3. $\log Z_\lambda$ is a convex function of λ, and strictly convex unless there exists a vector $\overrightarrow{\alpha} = (\alpha_1, \cdots, \alpha_c) \neq 0$ and a constant β such that
$$\sum_{k=1}^{c} \alpha_k \mathcal{E}_k(X) = \beta \ \ \forall\, x$$
(Remark: In this latter case we can simply drop one of the equations, $\mathcal{E}_l(x)$, without changing the set of functions $p_x = p_x(\vec{\lambda})$ defined in $(*)$.)

4. $\log Z_\lambda - \sum_{k=1}^{c} \lambda_k \theta_k$ is convex in λ and minimum when
$$E_\lambda[\mathcal{E}_k(X)] = \theta_k \ \ \forall\, k$$

Proof Start with 2 helpful identities

(a) $\dfrac{\partial}{\partial \lambda_k} Z_\lambda = \dfrac{\partial}{\partial \lambda_k} \sum_x h_x e^{\sum_l \lambda_l \mathcal{E}_l(x)} = \sum_x \mathcal{E}_k(x) h_x e^{\sum_l \lambda_l \mathcal{E}_l(x)} = Z_\lambda E_\lambda[\mathcal{E}_k(X)]$

(b) $\dfrac{\partial^2}{\partial \lambda_k \partial \lambda_l} Z_\lambda = \dfrac{\partial}{\partial \lambda_l} \sum_x \mathcal{E}_k(x) h_x e^{\sum_m \lambda_m \mathcal{E}_m(x)} = \sum_x \mathcal{E}_k(x) \mathcal{E}_l(x) h_x e^{\sum_m \lambda_m \mathcal{E}_m(x)} = Z_\lambda E_\lambda[\mathcal{E}_k(X) \mathcal{E}_l(X)]$

Now back to the proposition:

1. $\dfrac{\partial}{\partial \lambda_k} \log Z_\lambda = \dfrac{1}{Z_\lambda} \dfrac{\partial}{\partial \lambda_k} Z_\lambda \overset{(a)}{=} E_\lambda[\mathcal{E}_k(X)]$

2. $\dfrac{\partial^2}{\partial \lambda_k \partial \lambda_l} \log Z_\lambda = \dfrac{\partial}{\partial \lambda_l} \Big(\dfrac{\frac{\partial}{\partial \lambda_k} Z_\lambda}{Z_\lambda} \Big)$
$$= \frac{(\frac{\partial^2}{\partial \lambda_k \partial \lambda_l} Z_\lambda) Z_\lambda - (\frac{\partial}{\partial \lambda_k} Z_\lambda)(\frac{\partial}{\partial \lambda_l} Z_\lambda)}{Z_\lambda^2}$$
$$\overset{\text{(use a \& b)}}{=} E_\lambda[\mathcal{E}_k(X)\mathcal{E}_l(X)] - E_\lambda[\mathcal{E}_k(X)] E_\lambda[\mathcal{E}_l(X)]$$
$$= E_\lambda[(\mathcal{E}_k(X) - E_\lambda[\mathcal{E}_k(X)])\,(\mathcal{E}_l(X) - E_\lambda[\mathcal{E}_l(X)])]$$
$$= \mathbb{C}_\lambda[\mathcal{E}_k(X), \mathcal{E}_l(X)]$$

3. Let H be the Hessian of $\log Z_\lambda$, and write $\mathbb{V}[Y]$ for the variance of a random variable Y. For any $v \in \mathbb{R}^c$, $v^T H v = \sum_{k,l} v_k v_l \, \mathbb{C}_\lambda[\mathcal{E}_k(X), \mathcal{E}_l(X)] = \mathbb{V}_\lambda[\sum_k v_k \mathcal{E}_k(X)] \geq 0 \Rightarrow 2^{nd}$ derivative of $\log Z_\lambda \geq 0$ in every direction $\Rightarrow \log Z_\lambda$ is convex

   What is more, if $\mathbb{V}_\lambda[\sum_k v_k \mathcal{E}_k(X)] > 0$, then $\log Z_\lambda$ strictly convex.

   (Remark: If $\mathbb{V}_\lambda[\sum_k v_k \mathcal{E}_k(X)] = 0$, then $\sum_k v_k \mathcal{E}_k(x)$ is a constant, and we can drop at least one of the constraint functions, as noted earlier)

4. $\log Z_\lambda - \lambda \cdot \theta$ has same Hessian as $\log Z_\lambda$, and hence is also convex. To get its (global) minimum, set $\frac{\partial}{\partial \lambda_k} = 0 \ \forall \, k$:

$$\frac{\partial}{\partial \lambda_k}(\log Z_\lambda - \lambda \cdot \theta) \overset{\text{use } 1}{=} E_\lambda[\mathcal{E}_k(X)] - \theta_k = 0$$

   which completes the proof.

Let θ represents the vector $(\theta_1, \cdots, \theta_c)$. According to part 4 of the proposition, our task is to find the minimum, w.r.t. λ, of $\log Z_\lambda - \lambda \cdot \theta$, which is the convex function of λ. What's more, according to part 1 of the proposition, $\dfrac{\partial}{\partial \lambda_k}(\log Z_\lambda - \lambda \cdot \theta) = E_\lambda[\mathcal{E}_k(X)] - \theta_k$.

There are many ways to find the minimum of a convex function, most of which have simple software packages that can be enlisted. But instead of using a package we are going to "role our own"—by simply following the gradient, downhill.

Initialize $\lambda = \lambda^{(0)}$ and for $t = 1, 2, \cdots$, and some small $\varepsilon > 0$ iterate

$$\begin{aligned} \lambda^{(t+1)} &= \lambda^{(t)} - \varepsilon \nabla(\log Z_\lambda - \lambda \cdot \theta) \\ &= \lambda^{(t)} - \varepsilon \begin{pmatrix} E_{\lambda(t)}[\mathcal{E}_1(X)] \\ \vdots \\ E_{\lambda(t)}[\mathcal{E}_c(X)] \end{pmatrix} + \varepsilon\theta \end{aligned}$$

and then stop when $\|\lambda^{(t+1)} - \lambda^{(t)}\|$ is "small."

Special case: linear inequality constraints

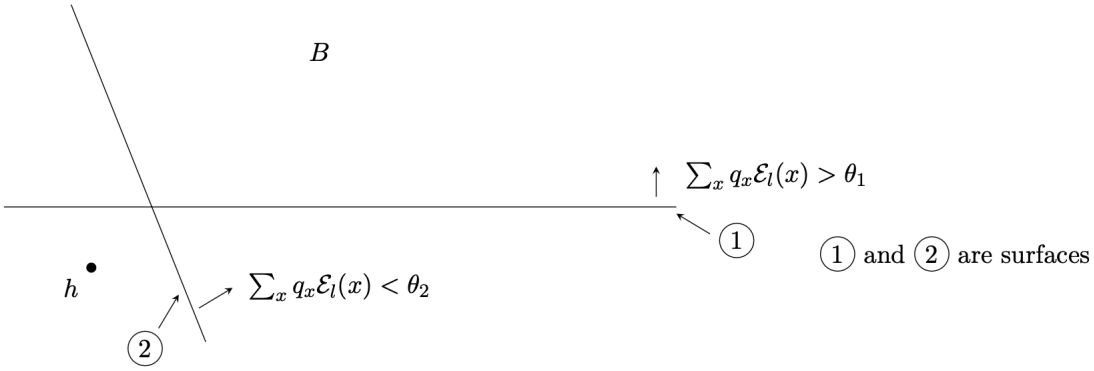
Suppose $B = \{q : \sum_x q_x \mathcal{E}_k(x) \, \square \, \theta_k, \forall \, k = 1, \cdots, c\}$, where for each k, $\square$ represents $<$ or $>$.

In this case, $p = \underset{q \in B}{\operatorname{argmin}}(D(q\|h))$ is unique. (Why? because (1) B is a convex set, and (2) if $D(\tilde{p}\|h) = D(p\|h)$, where p and $\tilde{p}$ are distinct and both on the boundary of B, then by the convexity of $D(q\|h)$:

$$D(\frac{p+\tilde{p}}{2}\|h) < \frac{1}{2}D(p\|h) + \frac{1}{2}D(\tilde{p}\|h) = D(p\|h)$$

which would contradict the fact that any minimum of $D(p\|h)$ must occur on the boundary.)

Consider two linear inequality constraints:



$p = \underset{q \in B}{\text{argmin}} D(q\|h)$ could fall on (1) or (2) or the intersection (depending on the location of h).

This is a "convex optimization problem" and there are many well-known methods for its solution.

Here is a simple approach for this particular situation:

(i) Solve for p using the single constraint

$$\sum_x q_x \mathcal{E}_1(x) > \theta_1$$

If the solution satisfies

$$\sum_x q_x \mathcal{E}_2(x) < \theta_2$$

then you are done (can you see why?)

(ii) Otherwise do (i), but with the two constraints in opposite order: Solve

$$\sum_x q_x \mathcal{E}_2(x) < \theta_2$$

If the solution satisfies

$$\sum_x q_x \mathcal{E}_1(x) > \theta_1$$

then you are done

(iii) Otherwise the solution satisfies both constraints and hence can be solved using

$$\Omega = \{q \in \mathbb{S} : \sum_x q_x \mathcal{E}_1(x) = \theta_1 \text{ and } \sum_x q_x \mathcal{E}_2(x) = \theta_2\}$$

I Gibbs ensembles and a few of their consequences

C Shannon's point of view

1. Typicality
2. Source Coding and the Kraft McMillan Inequalities
3. Optimal Codes
4. Misspecified Probability and the Relative Entropy $D(p\|q)$

Convention: throughout this section, $\log x$ *will mean* $\log_2 x$*.*

1 Typicality

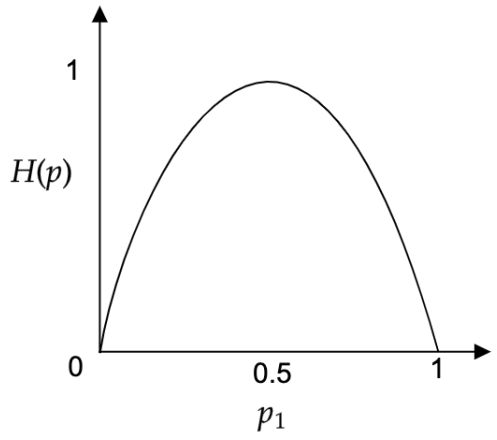
Let's flip a coin n times, where $p_1 = \mathbb{P}\{H\}$ and $p_2 = \mathbb{P}\{T\} = 1 = p_1$. Consider a "typical" sequence of outcomes

$$x_1, x_2, \ldots, x_n \quad \text{where} \quad x = \begin{cases} 1 & \text{for heads} \\ 2 & \text{for tails} \end{cases}$$

Then

$$\begin{aligned} \mathbb{P}\left(X_{1:n} = x_{1:n}\right) &= p_1^{n_1} p_2^{n_2} \\ &= p_1^{n\hat{p}_1} p_2^{n\hat{p}_2} \\ &\approx p_1^{np_1} p_2^{np_2} \qquad \text{(just the law of large numbers)} \\ &= 2^{n(p_1 \log p_1 + p_2 \log p_2)} \\ &= 2^{-nH(p)} \quad \text{(where } H(p) = -p_1 \log p_1 - p_2 \log p_2\text{, as usual, but in base 2 instead of } e\text{)} \end{aligned}$$

This is a little surprising, since it says that typical sequences all have about the same probability, $2^{-nH(p)}$, which means that there can only be, at most, $2^{nH(p)}$ typical sequences. Unless the coin is fair, $H(p)$ will be strictly less than one:



To the extent that $H(p)$ is less than 1, the number of typical sequence is exponentially smaller than 2^n, which is the number of possible sequences. Collectively, all of the other sequences are very unlikely. Which

suggests that we can find an encoding of sequences that will typically use fewer than n bits. In fact, we will see how to build codes that use fewer than $H(p) + \varepsilon$ bits, per symbol, on average, for any source distribution p on any number of symbols $\{1, 2, \ldots, s\}$.

2 Source Coding and the Kraft McMillan Inequalities

Here are two ways to think about the consequences of typicality in coin flips:

(i) The "information" per variable is $H(p)$ bits rather than 1 bit.

(ii) Consider the game of "20 questions": to learn the exact sequence $X_{1:n}$ you should only need about $nH(p)$ (binary) questions, on average. Compare these binary questions: "Is $\sum_1^k X_k$ even or odd?" versus "Does the sequence have any zeros?" The second question is wasteful. If the coin is anything but absurdly unbalanced, you will learn less than one bit of "information."

We will develop these ideas formally, starting with some basic notation:

$X \in \{1, 2, \cdots, s\} \sim p$ "source" (e.g. $s = 2^n$)

$C : \{1, 2, \cdots, s\} \to \{0, 1\}^*$ the set of finite nonempty strings of 0's & 1's, C is a "source code".

$|C(x)|$ length of the codeword $C(x)$, $x \in \{1, 2, \cdots, s\}$

Definition: A prefix code (aka instantaneous code) is a code C for which $C(x)$ is not a prefix of $C(\tilde{x})$ for any $x \neq \tilde{x}$, $x, \tilde{x} \in \{1, 2, \cdots, s\}$.

e.g. x	$C(x)$	$C'(x)$
1	0	0
2	1	10
3	00	11

$C(x)$ is not a prefix code and $C'(x)$ is a prefix code.

Virtue of a prefix code: uniquely decodable, left-to-right

e.g. 00010111010 decodes to 1112322 under C', but has 3 decodings under C.

Remark: there are many ways for a code to be uniquely decodable (e.g. a suffix code, which is right-to-left decodable). But for any uniquely decodable code C, you can always find a prefix code C' such that $|C'(x)| = |C(x)| \; \forall \, x \in \{1, 2, \cdots, s\}$ (see HW4 for a proof).

A key result in source coding is the Kraft-McMillan Inequality:

Every prefix code C satisfies

$$\sum_{x=1}^{s} 2^{-|C(x)|} \leq 1$$

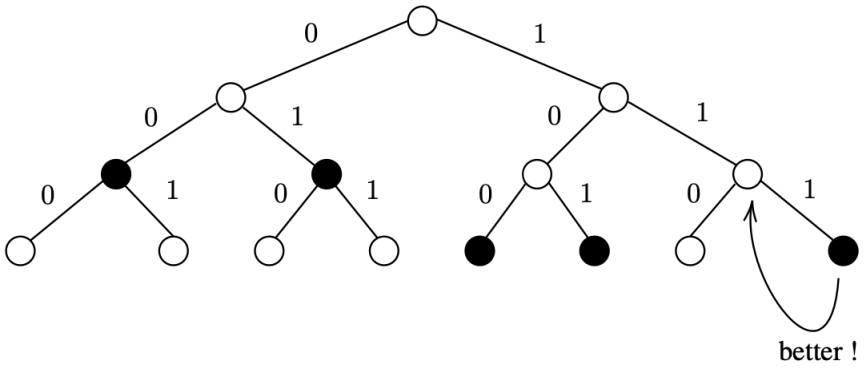
and for any set of code lengths $l_1, l_2, \cdots, l_s$ satisfying

$$\sum_{x=1}^{s} 2^{-l_x} \leq 1$$

there exists a prefix code C with $|C(x)| = l_x$, $x = 1, 2, \cdots, s$.

Remarks:

1. These and other related results are best understood through a picture: the "coding tree"



e.g.

x	$C(x)$
1	00
2	01
3	100
4	101
5	111

   A code satisfies the prefix condition $\Leftrightarrow$ no codeword has an ancestor codeword

2. If C is to be of any use, a minimal condition would be unique decodability: if

   $$C(x_1)C(x_2)\cdots C(x_i) = C(y_1)C(y_2)\cdots C(y_j) \quad \text{(products indicate concatenations)}$$

   then $i = j$ & $x_k = y_k \ \forall k$

3. The Kraft-McMillan results still hold for uniquely decodable codes.

   For example, $\{11, 011, 0011, 1110\}$ is neither prefix nor suffix, but it is, nevertheless, uniquely decodable:

starts with	1^{st} codeword
$011\cdots$	011
$0011\cdots$	0011
$110\cdots$	11
$1110\cdots$	1110
$11110\cdots$	11
$11111\cdots$	11

   (see Sardinas-Patterson algorithm for a method of determining whether C is uniquely decodable in polynomial time)

   The important point is that K-M inequality still holds with "prefix" replaced by "uniquely decodable", though the proof is harder and less transparent.

3 Optimal Codes

There are two important corollaries of the K-M inequality.

Corollary 1 (optimal codes)

Let $C^* = \underset{C \text{ prefix}}{\operatorname{argmin}} E|C(x)|$. Then

$$H(p) \leq E(|C^*(x)|) < H(p) + 1$$

Proof

1. $H(p) \leq E(|C^*(x)|)$
   It's enough to show $H(p) \leq E(|C(x)|)$ for an arbitrary fixed prefix code C. For every $x \in \{1, \ldots, s\}$, let $l_x = |C(x)$, $Z = \sum_x 2^{-l_x}$, and $q_x = \frac{1}{Z} 2^{-l_x}$. Since C is prefix, $Z \leq 1$. Then

$$\begin{aligned} E|C(x)| - H(p) &= \sum_x p_x l_x + \sum_x p_x \log p_x \\ &= \sum_x p_x \log 2^{l_x} + \sum_x p_x \log p_x \\ &= \sum_x p_x \log(\frac{p_x}{2^{-l_x}}) \\ &= \sum_x p_x \log(\frac{p_x}{\frac{2^{-l_x}}{Z} Z}) \\ &= \sum_x p_x \log(\frac{p_x}{q_x Z}) \\ &= D(p\|q) + \log \frac{1}{Z} \\ &\geq 0 + \log 1 = 0 \end{aligned}$$

   since $D(p\|q) \geq 0$ and, by the K-M inequality, $\sum_w 2^{-lw} \leq 1$.

2. $E(|C^*(x)|) < H(p) + 1$

   Let $l_x = \lceil \log \frac{1}{p_x} \rceil$. Then

$$\sum \frac{1}{2^{l_x}} \leq \sum \frac{1}{2^{\log \frac{1}{p_x}}} = \sum p_x = 1$$

   $\Rightarrow$ (by K-M) there exists a prefix code C with $|C(x)| = l_x$

   $\Rightarrow E|C^*(x)| \leq E|C(x)| = \sum p_x \lceil \log \frac{1}{p_x} \rceil \leq \sum p_x (\log \frac{1}{p_x} + 1) = H(p) + 1$

   One notable message from this proof is that we can get nearly optimal codes by using code lengths with approximately $l_x = \log \frac{1}{p_x}$ bits for each symbol x.

Corollary 2 (Block coding)

Let $X_{1:n} \sim$ iid p on $\{1, 2, \cdots, s\}$, and let

$$C_n^* = \underset{C_n \text{ on } x_{1:n}}{\operatorname{argmin}} E|C(X_{1:n})|$$

Then $H(p) \leq \frac{1}{n} E|C_n^*(X_{1:n})| \leq H(p) + \frac{1}{n}$.

(So we can get arbitrarily close to $H(p)$ bits per symbol, provided that we code large enough blocks of symbols.)

Proof: For the purpose of the proof, we can avoid some confusion by writing $p(x)$ instead of p_x.

We start with a useful and intuitive Lemma: If $X \sim p$ and $Y \sim q$ are independent, then $H(X,Y) = H(X) + H(Y)$, where $H(X,Y)$ refers to the entropy of the joint distribution, $p(x)p(y)$ on X and Y, and $H(X)$ and $H(Y)$ refer to the individual entropies $H(X)$ and $H(Y)$ respectively.

To prove the lemma, we just have to compute $H(X,Y)$:

$$\begin{aligned} H(X,Y) &= -\sum_x \sum_y p(x)q(y) \log p(x)q(y) \\ &= -\sum_x \sum_y p(x)q(y) \log p(x) - \sum_x \sum_y p(x)q(y) \log q(y) \\ &= -\sum_x p(x) \log p(x) - \sum_y q(y) \log q(y) = H(X) + H(Y) \end{aligned}$$

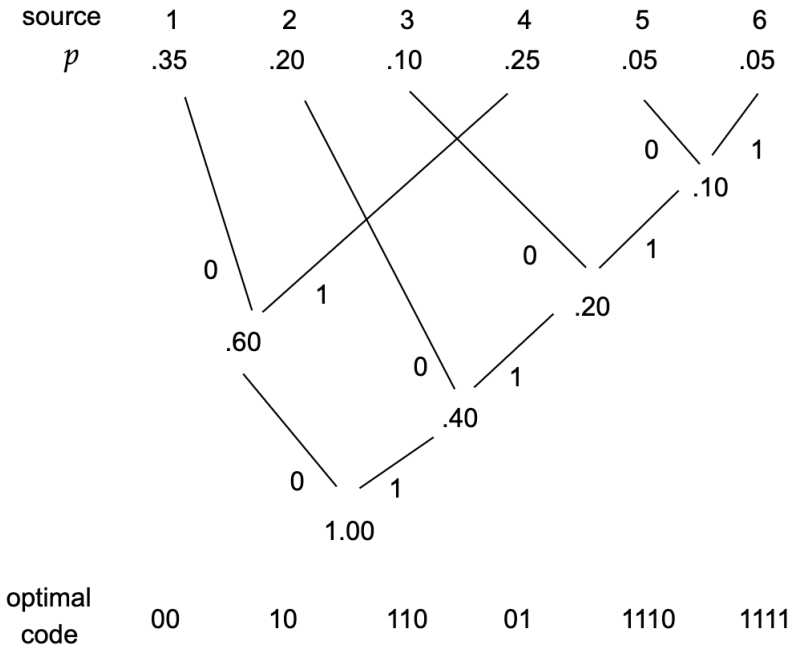
This makes sense: if X and Y are independent, then you learn $H(X)$ bits from observing X and an additional $H(Y)$ bits from observing Y.

Now let $p^{(n)}(X_{1:n}) = \prod_1^n p_(x_i)$ and apply the lemma, recursively, to X_m and $Y = X_{1:m-1}$, and conclude that $H(p^{(n)}) = nH(p)$.

Finally

$$\begin{aligned} H(p^{(n)}) \leq & E|C^*(X_{1:n})| < H(p^{(n)}) + 1 \qquad \text{(Corollary 1)} \\ \Rightarrow nH(p) \leq & E|C^*(X_{1:n})| < nH(p^{(n)}) + 1 \\ \Rightarrow H(p) \leq & \frac{1}{n} E|C^*(X_{1:n})| < H(p^{(n)}) + \frac{1}{n} \end{aligned}$$

But how do we actually construct an optimal code, C^*? This is the problem that the Huffman code solves. The proof is not hard, but we won't bother with it here. As for the description, by far the best way to learn the construction is from an example:



Huffman code, by example. Recursively combine the two most unlikely nodes, and then label each of the two branches, one with a zero and the other with a one. When the combined node has mass one, read off the code words by following the labelled path back to each of the source symbols. (Huffman codes are not unique since ties can be broken arbitrarily.)

Example of achieving the entropy lower bound
$s = 3$, $p_1 = \frac{1}{2}$, $p_2 = p_3 = \frac{1}{4}$

$H(p) = \frac{1}{2}\log 2 + \frac{2}{4}\log 4 = 1.5$

no advantage to use block coding

x	$p(x)$	$C(x)$
1	$\frac{1}{2}$	0
2	$\frac{1}{4}$	10
3	$\frac{1}{4}$	11

$$E|C(x)| = \frac{1}{2}\cdot 1 + \frac{1}{4}\cdot 2 + \frac{1}{4}\cdot 2 = 1.5 = H(p)$$

Example of achieving the entropy lower bound asymptotically
$s = 3$, $p_1 = p_2 = p_3 = \frac{1}{3}$

$H(p) = \log 3 \approx 1.5850$

$\underline{n=1}$	x	$p(x)$	$C(x)$
	1	$\frac{1}{3}$	0
	2	$\frac{1}{3}$	10
	3	$\frac{1}{3}$	11

$E|C(x)| = \frac{5}{3} \approx 1.6667 > H(p)$

Try block coding: $\underline{n=2}$

x	$p(x)$	$C(x)$
11	$\frac{1}{9}$	000
12	"	001
13	"	010
21	"	011
22	"	100
23	"	101
31	"	110
32	"	1110
33	"	1111

$$E|C(x)| = \frac{7}{9} \cdot 3 + \frac{2}{9} \cdot 4$$
$$= 3.2222\ldots \text{ total bits}$$
$$= 1.6111\ldots \text{ bits per symbol}$$

Each of these codes (for $n = 1$ and $n = 2$) is a Huffman code, and hence optimal. But the entropy lower bound is only achieved asymptotically:

$n = 1 \Rightarrow \cdots \approx 1.6667$ bits per symbol

$n = 2 \Rightarrow \cdots \approx 1.6111$ bits per symbol

$\vdots$

approaches $H(p) = 1.5850$ bits per symbol

4 Misspecified Probability and the Relative Entropy $D(p\|q)$

Another way to look at entropy, based on code lengths:

$$E|C^*(x)| = \sum_x p_x l_x^* \geq \sum_x p_x \log \frac{1}{p_x} \quad = H(p)$$

The optimal code will use about $\log \frac{1}{p_x}$ bits for symbol x.

Suppose that the source probability is p, but we believe, incorrectly, that the probability is q. Then we would use approximately $\log \frac{1}{q_x}$ bits for symbol x.

How many extra bits, on average, does the mistake cost us? About

$$\approx \sum_x p_x (\log \frac{1}{q_x} - \log \frac{1}{p_x}) = \sum_x p_x \log(\frac{p_x}{q_x}) = D(p\|q)$$

Interpretation: $D(p\|q)$ is the average extra bits per symbol that we need when we use the wrong distribution q, instead of the true source distribution p.

I Gibbs ensembles and a few of their consequences

D Symmetries and asymptotic independence

In another version of the Gibbs thought experiment we could consider an *isolated* system of N ideal (and identical) gas molecules, as opposed to a system in contact with an infinite heat bath. The key difference is that there would be no fluctuations in the kinetic energy, since the system is isolated and therefore not exchanging energy with its surroundings (by assumption, all of the system's energy is kinetic energy). In this case, the Gibbs (maximum-entropy) distribution is replaced by the *uniform* distribution on all configurations of the system that have a fixed and constant kinetic energy, i.e.

$$\sum_{i=1}^{N} \left(v_x(i)^2 + v_y(i)^2 + v_z(i)^2 \right) = \theta$$

for a suitable constant θ that will be proportional to N as well as the temperature. (The dependency on N comes from the fact that temperature measures the *average* rather than the *total* kinetic energy.) Equivalently, by simply relabeling the velocity components we can write this as $\sum_{k=1:n} x_k^2 = \theta$, where $n = 3N$.

Since temperature is a measure of the average kinetic energy, and since we will be interested in taking the limit as the number of particles grows, we will take theta to be n, which will be correct up to a scaling which just depends on the units of temperature and kinetic energy:

$$\sum_{k=1}^{n} x_k^2 = n \tag{1}$$

Our interest is in the uniform distribution on all configurations that satisfy (1). Formally,

$$X_{1:n} \ \sim \ \text{Uniform on } \{x_{1:n} : \sum_{k=1:n} x_k^2 = n\}$$

or, more succinctly, $X_{1:n} \ \sim \ U\left(S^{n-1}(n^{1/2})\right)$, where $S^{n-1}(r)$ is the sphere with radius r in n dimensions. The superscript $n-1$ refers to the dimensionality of the surface and not the space in which the surface resides. (So, for example, $S^2(1)$ is the surface of the unit ball in $\mathbb{R}^3$.)

Under this distribution on the collection of velocities, what is the distribution on the individual velocities? You could think about it this way: imagine a spherical shell in $\mathbb{R}^n$, and pick one of the n axes, let's say the first. Suppose the shell were shattered into tiny pieces and each piece fell perpendicularly to the first axis. What would the histogram of masses look like? It would depend very much on n, but Maxwell (1831-1879) and Boltzmann (1844-1906) had a way of making a very excellent guess, by exploiting the symmetries of the problem.

Reason as follows:

(1) (Boltzmann and "molecular chaos") The number of particles in, say, a liter of the gas is very large (e.g. 10^{23} molecules). Intuitively, it follows that if I told you the value of one of the velocities (say X_i) it would tell you very little about the value of any other velocity (say X_j).

What's more, it is clear that each velocity, X_k, has mean zero ($E[X_k] = 0$) since the sphere is centered at the origin. Concerning the variance, $E[X_k^2]$:

$$\begin{aligned}
\sum_{k=1:n} X_k^2 = n \quad &\text{(from equation 1)} \quad \Rightarrow \\
E[\sum_{k=1:n} X_k^2] = n \quad &\Rightarrow \\
\sum_{k=1:n} E[X_k^2] = n \quad &\Rightarrow \\
E[X_k^2] = 1 \ \ \forall k \quad &\text{(symmetry of the sphere—every axis is the same)}
\end{aligned}$$

(2) Consequently, for any $m << n$ (m could be large, but just a small fraction of n), $X_{1:m}$ will be approximately independent and identically distributed, with common mean zero and common variance 1.

(3) Hence, by the central limit theorem

$$\frac{1}{\sqrt{m}} \sum_{k=1}^{m} X_k \approx \mathcal{N}(0,1) \qquad \text{(central limit theorem)}$$

or, equivalently,

$$\left(\frac{1}{\sqrt{m}}, \frac{1}{\sqrt{m}}, \ldots, \frac{1}{\sqrt{m}}, 0, 0, \ldots, 0\right) \cdot (X_1, X_2, \ldots, X_n) \approx \mathcal{N}(0,1)$$

where the left-hand expression defines the n dimensional vector with the first m components equal to $\frac{1}{\sqrt{m}}$ and the last $n - m$ equal to zero.

(4) But the uniform distribution on $S^{n-1}(\sqrt{n})$ is rotationally invariant. Hence, if the projection of $X_{1:n}$ onto the unit-length vector $\vec{v} = \left(\frac{1}{\sqrt{m}}, \frac{1}{\sqrt{m}}, \ldots, \frac{1}{\sqrt{m}}, 0, 0, \ldots, 0\right)$ is approximately standard normal, then so is the projection onto any other unit-length vector, e.g. the unit vector $\vec{v} = (1, 0, 0, \ldots, 0)$. In other words

$$X_1 = (1, 0, 0, \ldots, 0) \cdot X_{1:n} \ \approx \ \mathcal{N}(0,1)$$

In fact, it turns out to be true that the components of the velocities of a gas, under the assumed conditions, do indeed behave like independent normal random variables. So, for example, the speed (in the proper units) of a randomly selected molecule has the distribution $\sqrt{v_x(i)^2 + v_y(i)^2 + v_z(i)^2} \sim \sqrt{Z_1^2 + Z_2^2 + Z_3^2}$ where $Z_1, Z_2, Z_3 \ \sim \ iid \ \mathcal{N}(0,1)$. This is called the Maxwell (sometimes Maxwell-Boltzmann) velocity distribution.

Remarks:

1. Formally, the argument leads to the following conjecture: If $X_{1:n} \ \sim \ U\left(S^{n-1}(n^{1/2})\right)$ then for every fixed integer $m \geq 1$ and every set[1] $A \subseteq \mathbb{R}^m$,

$$\mathbb{P}\left(X_{1:m} \in A\right) \stackrel{n\to\infty}{\longrightarrow} \mathbb{P}\left(Z_{1:m} \in A\right)$$

where $Z_{1:m} \ \sim \ iid \ \mathcal{N}(0,1)$.

[1] Actually, every Borel set, but this is a minor point and, besides, it's safe to ignore it if you haven't yet learned measure theory.

2. The conjecture is true, and is proven in two different ways in HW4 for the special case $m = 1$. Those same proofs easily extended to general m.

3. Keep in mind that $X_{1:n}$ are certainly not independent. After all, they must respect the deterministic constraint $\sum_{k=1:n} X_k^2 = n$.

4. The equivalent result, starting with the uniform distribution on the unit sphere, $(Y_1, \ldots, Y_n) \sim U\left(S^{n-1}(1)\right)$, states that the distribution of $\sqrt{n}(Y_1, \ldots, Y_m)$ converges to $iid\ \mathcal{N}(0, 1)$. (Just define $Y_k = X_k/\sqrt{n}$ for each $k = 1 : n$.)

5. The random variables $X_1, \ldots, X_n$ are called "exchangeable" since any permutation of their ordering leaves the joint distribution unchanged. There are some remarkable general theorems about exchangeable random variables, perhaps the most beautiful being de Finetti's Theorem: Every infinite exchangeable sequence of random variables is a mixture of independent and identically distributed random variables.

II Some Statistical Theory

* focus on nonparametrics

* cool tricks (support vector machine, cross validation, $\cdots$)

* cautionary notes ("bias-variance dilemma", "curse of dimensionality")

* methods that find their way into applications (neural nets, recordings from real neurons, image analysis, images processing, comp. biology, $\cdots$)

A Estimation

1. The estimation problem
2. Performance measures
3. Kernel density estimation
4. Cross validation
5. Maximum likelihood estimation

1 The estimation problem

$X_1, \cdots \sim iid\ P_\theta$, θ unknown

Goal: Estimate θ using $X_1, \cdots, X_n$

$\theta \in \mathbb{R}^d$ $(d < \infty)$ "parametric estimation"

$\theta \in \mathbb{R}^\infty$ "nonparametric estimation"

Generically, write: $\hat{\theta} = \hat{\theta}_n = \hat{\theta}_n(X_{1:n})$

(i) $\mathbb{E}(\hat{\theta}) - \theta$: is the "bias" of the estimator (if bias = 0, "unbiased")

(ii) if $\hat{\theta}_n \longrightarrow \theta$: then we call $\hat{\theta}_n$ "consistent"

But what exactly does $\hat{\theta}_n \longrightarrow \theta$ mean?

Modes of convergence: If $\theta, \hat{\theta}(X_{1:n}) \in \mathbb{R}^d$:

(i) $\hat{\theta}_n \to \theta$ almost surely if

$$\mathbb{P}(\hat{\theta}(X_{1:n}) \to \theta) = 1$$

(ii) $\hat{\theta}_n \to \theta$ in probability if

$$\mathbb{P}(|\hat{\theta}(X_{1:n}) - \theta| > \varepsilon) \xrightarrow{n\to\infty} 0 \quad \forall \varepsilon > 0$$

(iii) $\hat{\theta}_n \to \theta$ in mean square if

$$\mathbb{E}[|\hat{\theta}(X_{1:n}) - \theta|^2] \xrightarrow{n\to\infty} 0$$

There are many other modes of convergence. These are some of the more common ones.)

e.g. $\mu = \mathbb{E}[X]$ and

$$\hat{\mu}_n(X_{1:n}) = \bar{X} \doteq \frac{1}{n}\sum_{k=1}^{n} X_k$$

Then $\hat{\mu}_n$ is unbiased and consistent in most senses (under mild conditions on P)

e.g. $X \sim \mathrm{N}(\mu, \sigma^2)$, $\hat{\mu} = \overline{X}$, and $S_n^2 = \frac{1}{n-1}\sum_{k=1}^{n}(X_k - \overline{X})^2$

Then both are unbiased and consistent.

But why, intuitively, does S_n^2 require normalization by $n-1$ instead of n in order to be unbiased? Evidently, $\frac{1}{n}\sum_{k=1}^{n}(X_k - \overline{X})^2$ would be too small. Here's why:

$$\sigma^2 = \mathbb{E}[(X-\mu)^2] = \mathbb{E}[\frac{1}{n}\sum_{1}^{n}(X_k - \mu)^2]$$

$$\geq \mathbb{E}[\inf_{\alpha} \frac{1}{n}\sum_{1}^{n}(X_k - \alpha)^2] = \mathbb{E}[\frac{1}{n}\sum_{1}^{n}(X_k - \overline{X})^2]$$

(with equality iff $\bar{X} = \mu$).

(If you want to see why $\frac{1}{n} \to \frac{1}{n-1}$ is *exactly* the right correction, then you will need to calculate $\mathbb{E}[\sum_{1}^{n}(X_k - \overline{X})^2]$ directly, which is not difficult.)

e.g. $X \sim P$, (so $\theta = P_\theta$) "nonparametric"

The natural estimator is the empirical distribution:

$$\hat{P}_n = \hat{P}(X_{1:n}) = \frac{1}{n}\sum_{1}^{n}\delta_{X_k} \quad \text{(where } \delta_{X_k} \text{ is the dist. with all of its mass at } X_k)$$

Equivalently, for every $B \subset \mathbb{R}$ ("measurable") define

$$\hat{P}_n(B) = \frac{1}{n}\sum_{1}^{n}\mathbb{1}_{X_k \in B}$$

Then $\hat{P}_n(B)$ is unbiased and consistent for $P(B)$ for every such B.

In fact, $\hat{P}_n(B)$ is even uniformly consistent for a large class of B's, namely $\{B : B = (-\infty, x] \text{ for some } x \in \mathbb{R}\}$:

$$\sup_x |\hat{P}_n(-\infty, x) - P(-\infty, x)| \xrightarrow{n\to\infty} 0 \quad \text{a.s}$$

(this is the Glivenko-Cantelli lemma)

But don't push your luck: suppose P is continuous (e.g. $X \sim \mathrm{N}(0,1)$), then for every n

$$P(\bigcup_{1}^{n}\{X_k\}) = 0 \quad \text{but} \quad \hat{P}(\bigcup_{1}^{n}\{X_k\}) = 1$$

So there is always a B for which $P(B) = 0$ but $\hat{P}(B) = 1$. Hence, we certainly cannot say that

$$\sup_B |\hat{P}_n(B) - P(B)| \xrightarrow{n\to\infty} 0$$

2 Performance measures

We will focus on mean-squared errors:

Mean Squared Error (MSE) for parametric estimators

Given $X_{1:n} \sim iid\ P_\theta, \theta \in \mathbb{R}$

$$\begin{aligned}
\mathrm{MSE}_\theta(\hat{\theta}_n) = \mathbb{E}_\theta[(\hat{\theta}_n - \theta)^2] & \\
&= \mathbb{E}_\theta[((\hat{\theta}_n - \mathbb{E}_\theta[\hat{\theta}_n]) + (\mathbb{E}_\theta[\hat{\theta}_n] - \theta))^2] \\
&= (\mathbb{E}_\theta[\hat{\theta}_n] - \theta)^2 + \mathbb{E}_\theta[(\hat{\theta}_n - \mathbb{E}_\theta[\hat{\theta}_n])^2] \\
&= (\mathbb{E}_\theta[\hat{\theta}_n] - \theta)^2 + \mathrm{Var}_\theta(\hat{\theta}_n) \\
&= \mathrm{Bias}^2 + \mathrm{Variance}
\end{aligned}$$

e.g. $X_{1:n} \sim iid$ with common mean μ and common variance σ^2

① $\hat{\mu}_n = \overline{X} \Rightarrow \mathbb{E}\,\mu_n \Rightarrow \mu \Rightarrow$ unbiased
$\mathrm{MSE}(\hat{\mu}_n) = \mathrm{Bias}^2 + \mathrm{Var} = \mathrm{Var}[\frac{1}{n}\sum_1^n X_k] = \frac{\sigma^2}{n} \to 0 \Rightarrow$ consistent

② $\hat{\sigma}_n^2 = \frac{1}{n}\sum_1^n (X_k - \hat{\mu}_n)^2$
$\Rightarrow \mathbb{E}[\hat{\sigma}_n^2] = \mathbb{E}(X_1 - \frac{1}{n}\sum_1^n X_k)^2 = \cdots = \frac{n-1}{n}\sigma^2$ (calculation is a bit messy)
$\Rightarrow \mathbb{E}[\hat{\sigma}_n^2] - \sigma^2 = -\frac{1}{n}\sigma^2$ (a small negative bias)

③ $S_n^2 = \frac{1}{n-1}\sum_1^n (X_k - \hat{\mu}_n)^2 = \frac{n}{n-1}\hat{\sigma}_n^2 \Rightarrow \mathbb{E}\,S_n^2 = \frac{n}{n-1}\,\mathbb{E}\,\hat{\sigma}_n^2 = \sigma^2$ (unbiased)

But both ② and ③ are mean square consistent:

$$\mathbb{E}\,|\hat{\sigma}_n^2 - \sigma^2|^2 \to 0 \text{ and } \mathbb{E}\,|\hat{S}_n^2 - \sigma^2|^2 \to 0$$

Which one should we prefer?

Suppose your criterion is MSE. The one with smaller MSE depends on the distribution on X, and this gets very messy.

Let's take the Gaussian case:

	$\hat{\sigma}_n^2$		S_n^2
bias	$-\frac{\sigma^2}{n}$		0
variance	$\frac{2(n-1)}{n^2}\sigma^4$		$\frac{2}{n-1}\sigma^4$
MSE	$\frac{2n-1}{n^2}\sigma^4$	$<$	$\frac{2}{n-1}\sigma^4$

The biased estimator is the better MSE choice (in this case!)

Remarks: for Gaussian

i. S_n^2 has lowest variance among all unbiased estimators of σ^2

ii. $\frac{n-1}{n+1}S_n^2$ has lowest MSE among all estimators of the form αS^2

Message: sometimes best to trade bias for variance or vice versa

MSE for pdf's

Let $\theta = f$ be an unknown density, with estimator $\hat{f}_n(x) = \hat{f}(x; X_{1:n})$

How close is $\hat{f}_n$ to f? One natural measure is the integrated square error:

$$\text{ISE} \triangleq \int_x (\hat{f}(x; X_{1;n}) - f(x))^2 dx \qquad \text{(a random variable!)}$$

Now define

$$\text{MISE} \triangleq \mathbb{E}[\text{ISE}] = \int_x \mathbb{E}[(\hat{f}(x; X_{1:n}) - f(x))^2] dx$$

We can decompose the integrand with the same reasoning as in the parametric case:

$$\begin{aligned}\mathbb{E}[(\hat{f}(x; X_{1:n}) - f(x))^2] &= \Big(\mathbb{E}[\hat{f}_n(x; X_{1:n})] - f(x)\Big)^2 + \mathbb{E}[\Big(\hat{f}_n(x; X_{1:n}) - \mathbb{E}[\hat{f}_n(x; X_{1:n})]\Big)^2] \\ &= \text{Bias}(x)^2 + \text{Var}(x)\end{aligned}$$

And hence

$$\text{MISE} = \int_x \text{Bias}(x)^2 + \int_x \text{Var}(x) dx$$

3 Kernel density estimation

Goal: estimate a density, $f(x)$, from $X_{1:n} \sim$ iid f

Method: Use a "kernel" function $k(x)$:

$$k \geq 0, \int k(x)dx = 1, \int xk(x)dx = 0, \int x^2 k(x)dx = 1$$

A common example is the standard normal:

$$k(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

Assume: $\int f(x)^2 dx < \infty$ and assume some smoothness of f and k $\cdots$

To control bias vs variance, introduce a "smoothing parameter" w:

$$k_w(x) \triangleq \frac{1}{w} k(\frac{x}{w})$$

in which case, for all $w > 0$

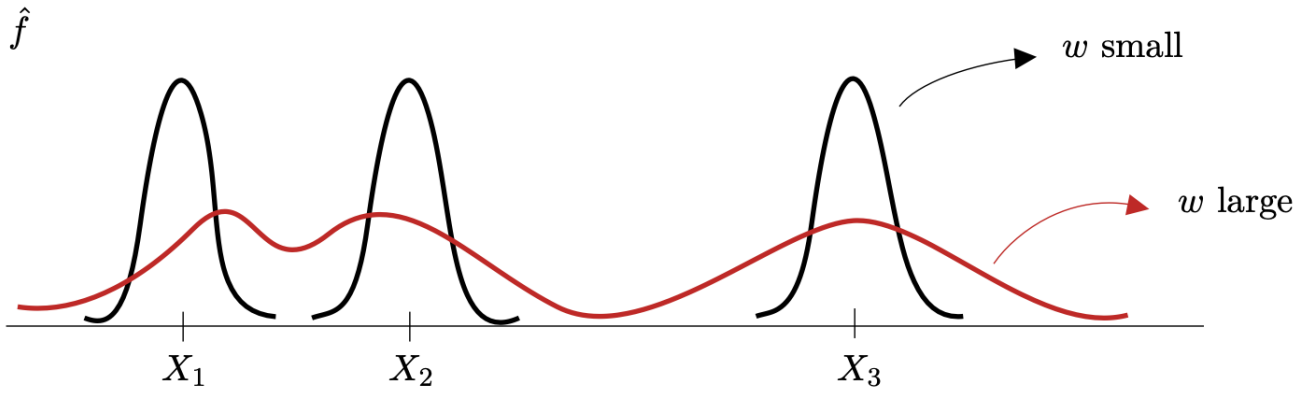
i. $\int k_w(x)dx = 1$

ii. $\int xk_w(x)dx = 0$

iii. $\int x^2 k_w(x)dx = w^2$

Here's the estimator

$$\hat{f} = \hat{f}_{n,w}(x) = \hat{f}_{n,w}(x; X_{1:n}) \triangleq \frac{1}{nw}\sum_{i=1}^{n} k(\frac{x - X_i}{w}) = \frac{1}{n}\sum_{i=1}^{n} k_w(x - X_i)$$



$$w \downarrow 0 : \quad \text{bias} \downarrow \quad \text{var} \uparrow$$
$$w \uparrow \infty : \quad \text{bias} \uparrow \quad \text{var} \downarrow$$

Let's analyze the bias and variance terms, formally, in terms of their dependence on the number of samples n and the bandwidth (aka smoothing parameter) w:

$$\text{MISE}_{n,w} = \int_x B_{n,w}(x)^2 + V_{n,w}(x)dx$$

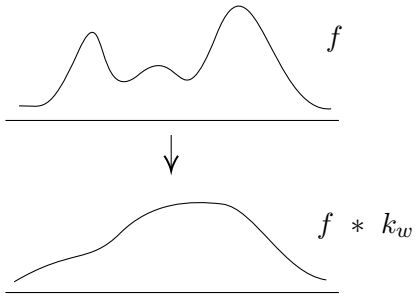
$\underline{B_{n,w}(x)}$

$$\begin{aligned}
B_{(n,w)}(x) &= \mathbb{E}[\hat{f}(x; X_{1:n})] - f(x) \\
\mathbb{E}[\hat{f}(x; X_{1:n})] &= \frac{1}{n}\sum_{k=1}^{n} \mathbb{E}[k_w(x - X_i)] \\
&= \mathbb{E}[k_w(x - X)] \\
&= \int_t f(t)k_w(x-t)dt \longrightarrow \begin{cases} f(x) & \text{as } w \to 0 \\ 0 & \text{as } w \to \infty \end{cases}
\end{aligned}$$

does not depend on $n \Rightarrow$ no hope for consistency unless $w \to 0$

Convolution! ("$f * k_w$") Think of it as the density of sum of $X \sim f$, $Y \sim k_w$, X, Y ind

smooths f



$$\Rightarrow B^2_{(n,w)}(x) \to \begin{cases} 0 & \text{as } w \to 0 \\ f(x)^2 & \text{as } w \to \infty \end{cases}$$

$\underline{V_{n,w}(x)}$

$$\begin{aligned}
V_{n,w}(x) &= \text{Var}[\hat{f}_{n,w}(x; X_{1:n})] \\
&= \text{Var}[\frac{1}{n}\sum_{1}^{n} k_w(x - X_i)] \\
&= \frac{1}{n}\text{Var}[k_w(x - X)] \\
&= \frac{1}{n}\,\mathbb{E}[k_w(x - X)^2] - \frac{1}{n}\,\mathbb{E}[k_w(x - X)]^2
\end{aligned}$$

Define $V^{(1)}_{n,w}(x) = \frac{1}{n}\,\mathbb{E}[k_w(x - X)^2]$ and $V^{(2)}_{n,w}(x) = \frac{1}{n}\,\mathbb{E}[k_w(x - X)]^2$.

$$V_{n,w}^{(1)}(x) = \frac{1}{n}\int_t f(t)\left(\frac{1}{w}k(\frac{x-t}{w})\right)^2 dt \longrightarrow \begin{cases}\infty & \text{as } w \to 0 \text{ (because of an "extra" } \frac{1}{w}) \\ 0 & \text{as } w \to \infty\end{cases}$$

$$V_{n,w}^{(2)}(x) = \frac{1}{n}\left(\int_t f(t)(k_w(x-t)dt\right)^2 \xrightarrow[\text{above}]{\text{as}} \begin{cases}\frac{1}{n}f(x)^2 & \text{as } w \to 0 \\ 0 & \text{as } w \to \infty\end{cases}$$

$$V_{n,w}(x) = V_{n,w}^{(1)}(x) - V_{n,w}^{(2)}(x)$$

$$\Rightarrow V_{n,w}(x) \to \begin{cases}\infty & \text{as } w \to 0 \\ 0 & \text{as } w \to \infty\end{cases}$$

What about $\text{MISE}_{(n,w)} = \int_x B_{(n,w)}(x)^2 + V_{(n,w)}(x)dx$? How can we get both the bias and the variance to go to zero?

Since the bias does not depend on n and will only go to zero if $w \to 0$, this is evidently a necessary condition for MISE consistency. Now If we take a closer look at each term as $w \to 0$, and assume some standard smoothness and integrability assumptions, then we can write the asymptotic behaviors more precisely as

Bias: $B_{(n,w)}^2 = O(w^4)$ as $w \to 0$ (i.e. there exists a constant α such that $B_{n,w}^2 \leq \alpha w^4$ for all w sufficiently small)

Variance: $V_{n,w} = O(\frac{1}{nw})$ as $w \to 0$ (i.e. there exists a constant β such that $V_{n,w} \leq \frac{\beta}{nw}$ for all n and all w sufficiently small)

Remarks

1. α and β depend on f, which is unknown (but not unlike $\hat{\mu}_n = \overline{X} \Rightarrow \mathbb{E}[(\hat{\mu}_n - \mu)^2] = \cdots = \frac{\sigma^2}{n}$ when σ^2 is unknown)
2. $\hat{f}(x)$ is MISE consistent if we choose $w = w_n$ such that

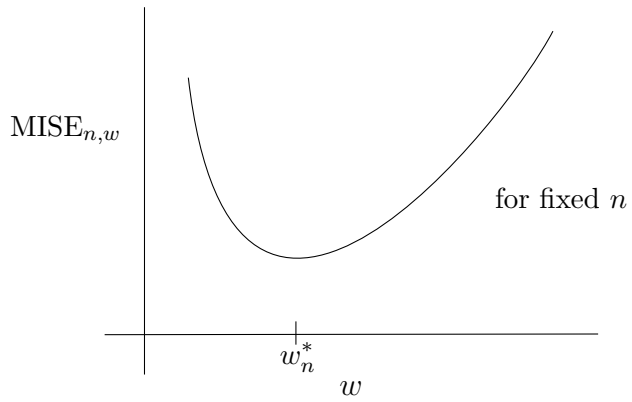
$$w_n \to 0 \text{ as } n \to \infty \text{ "kill the bias"}$$

$$nw_n \to \infty \text{ as } n \to \infty \text{ "kill the variance"}$$

3. For example, $\hat{f}(x)$ is MISE consistent if $w_n = \frac{1}{\sqrt{n}}$
4. Is there an optimal rate. Yes, at least in terms of the power of n: minimize over w

$$\frac{d}{dw}(\alpha w^4 + \frac{\beta}{nw}) = 0 \Rightarrow w_n^* = \frac{c}{n^{1/5}}$$

which balances bias and variance terms



5. $O(\frac{1}{n^{1/5}})$ is really slow, and of no use in practice!

6. In general finding a good Bias/Variance tradeoff can be tricky in nonparametric estimation

4 Cross validation

Cross validation is a very general approach to balancing bias and variance. It can fail badly, but oftentimes it works amazingly well. Here's an example for the kernel density estimator:

Suppose that $X_{1:n} \sim$ iid f, where f is an unknown pdf.

Given a kernel k, $k_w(x) = \frac{1}{w}k(\frac{x}{w})$ define

$$\hat{f}_w(x) = \hat{f}_{n,w}(x; X_{1:n}) = \frac{1}{n}\sum_{i=1}^{n} k_w(x - X_i)$$

The bandwidth w is critically important—but how do we choose w without already knowing f?

The idea is choose w to minimize an *estimate* of the ISE (hence the term "data-driven smoothing parameter"). Expanding out ISE leads to three integrals:

$$\text{ISE} = \int_x |\hat{f}_w(x) - f(x)|^2 dx =$$
$$\int_x \hat{f}_w(x)^2 dx - 2\int_x \hat{f}_w(x) f(x) dx + \int_x f(x)^2 dx$$

Call them I, II, and III, respectively.

For any given value of w, the first integral, I, can be computed directly by numerical integration, since it only involves the *known* function $\hat{f}_w(x)$.

The second integral is more problematic, since it depends on both w and the *unknown* function f. We will need to use the data (i.e. the observed values of $X_1, \ldots, X_n$) to approximate II.

As for III, we can ignore it. Our interest is in finding w that approximately minimizes ISE, whereas III has no dependence on w.

The key, then, is to devise an estimator for II:

$$\begin{aligned}
\text{II} &= \int_x \hat{f}_w(x) f(x) dx = \int_x \hat{f}_{n,w}(x; X_{1:n}) f(x) dx \\
&= \mathbb{E}_{X_{n+1}}[\hat{f}_{n,w}(X_{n+1}; X_{1:n})] \quad (X_{n+1} \text{ would be one additional observation}) \\
&\approx \mathbb{E}_{X_n}[\hat{f}_{n-1,w}(X_n; X_{1:n-1})] \quad (\text{don't expect much diff. between est. with } n \text{ vs. } n+1) \\
&\approx \frac{1}{n} \sum_{i=1}^{n} \hat{f}_{n-1,w}(X_i; {}_iX) \quad ({}_iX \text{ is } (X_1, \ldots, X_n), \text{ but without } X_i, \text{ "leave-one-out"}) \qquad (1)
\end{aligned}$$

Write $f^{(i)}_{n-1,w}$ for the estimator derived from the $n-1$ observations obtained by leaving out X_i:

$$f^{(i)}_{n-1,w}(x) = f^{(i)}_{n-1,w}(x; {}_i X) = \frac{1}{n-1} \sum_{\{j: j \neq i\}} k_w(x - X_j) \qquad (2)$$

in which case

$$\text{II} \approx \frac{1}{n} \sum_{i=1}^{n} f^{(i)}_{n-1,w}(X_i)$$

Since $f^{(i)}_{n-1,w}(x)$ is independent of X_i, we can think of $f^{(i)}_{n-1,w}(X_i)$ as a measure of evidence for the particular value w of the bandwidth; large values of $f^{(i)}_{n-1,w}(X_i)$ support the model (and hence the choice w) in the sense that the new observation X_i was anticipated by a high likelihood. Small values of $f^{(i)}_{n-1,w}(X_i)$ suggest the opposite—w is a poor choice for the bandwidth.

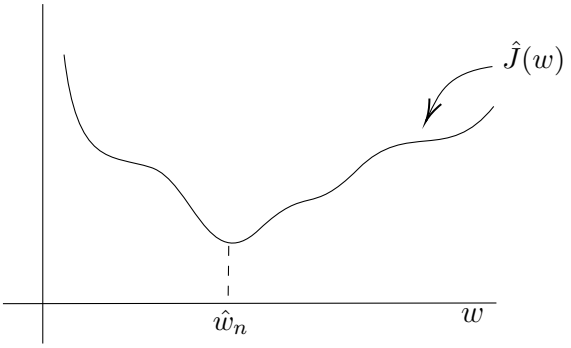
Define $J(w) = \int_x \hat{f}_w(x)^2 dx - 2 \int_x \hat{f}_w(x) f(x) dx$. Then

$$\operatorname*{argmin}_w \text{ISE} = \operatorname*{argmin}_w (\text{I-2II}) = \operatorname*{argmin}_w J(w) \approx \operatorname*{argmin}_w \hat{J}(w)$$

$$\text{where} \quad \hat{J}(w) \doteq \int_x \hat{f}_{n,w}(x; X_{1:n})^2 dx - \frac{2}{n} \sum_{i=1}^{n} \hat{f}^{(i)}_{n-1,w}(X_i)$$

which does not depend on f.

The hope is that $\hat{J}(w)$ will look something like this:



and then we could choose $\hat{w}_n = \operatorname{argmin} \hat{J}$ and use the estimator $\hat{f}_{\hat{w}_n} = \hat{f}_{n,\hat{w}_n}(x; X_{1:n})$ to estimate f. This would involve a lot of computation, but in light of a remarkable result by C. Stone, it might well be worth the effort:

Theorem (C. Stone, 1984)

$$\mathbb{P}\left(\frac{\mathrm{ISE}(\hat{f}_{\hat{w}_n}, f)}{\inf_w \mathrm{ISE}(\hat{f}_w, f)} \xrightarrow[n\to\infty]{} 1\right) = 1$$

This is almost too good to be true: using $\hat{w}_n$ asymptotically same performance as best possible w!

Cautions:

(1) Convergence can still be very slow

(2) Can be disasterously slow in high dimensions:

$$f(x) \to f(\vec{x}), \quad \vec{x} \in \mathbb{R}^d$$

$$k_w(x) \to k_w(\vec{x}) \triangleq \prod_{i=1}^{d} k_w(x_i)$$

Everything goes through, just as in the one-dimensional case discussed above. But good performance in high dimensions requires a lot more data than in one-dimension.

(3) For example, assume that $f(\vec{x}) \sim$ iid $N(0,1)$, in which case $f(0) = (\frac{1}{\sqrt{2\pi}})^d$. Now pretend like we don't know f and we use the (Gaussian) kernel estimator to estimate $f(0)$. Then the estimator will converge to $(\frac{1}{\sqrt{2\pi}})^d$, but how many samples would we need to get a good approximation with high confidence? Consider this experiment, by B. Silverman:

(4) (Silverman's cautionary tale) Consider using $\hat{f}_{n,w_n^*}(0; X_{1:n})$ to estimate $f(0)$, where w_n^* is the *true optimal* ISE bandwidth. (Which can be computed for each sample, since we already know f.) How close is $\hat{f}_{n,w_n^*}(0; X_{1:n})$ to $f(0)$? Specifically, for each dimension d consider the number of samples n needed to achieve $\mathbb{E}[|\hat{f}_{n,w_n^*}(0; X_{1:n}) - f(0)|^2] \le 0.1 f(0)$:

d	n
1	4
2	19
5	768
7	10700
10	842000

Nonparametric estimation can be very difficult, indeed, in high dimensions ("curse of dimensionaility").

5 Maximum likelihood estimation

Return to the generic estimation problem: $X \sim$ *iid* P_θ with θ unknown, and estimator $\hat{\theta} = \hat{\theta}_n(X_{1:n})$.

The most commonly used general approach to defining an estimator $\hat{\theta}$ is through "maximum likelihood":

$$\hat{\theta}_n(X_{1:n}) = \operatorname*{argmax}_\theta L(X_{1:n}; \theta)$$

where L, the likelihood function, is defined by

$$L(X_{1:n};\theta) = \prod_{k=1}^{n} p(X_k;\theta) \text{ (the joint distribution of the observerations)}$$

Here, p is a pmf if p is discrete, and p is pdf if p is continuous and has a density.

This makes good sense: choose the parameter to make the data as believable (likely) as possible.

There is an immediate connection to relative and cross entropies:

$$\begin{aligned}
\hat{\theta}_n(X_{1:n}) &= \operatorname*{argmax}_\theta \prod_{k=1}^{n} p(x_k;\theta) \\
&= \operatorname*{argmax}_\theta \frac{1}{n}\sum_{k=1}^{n} \log p(x_k;\theta) \\
&= \operatorname*{argmax}_\theta \sum_{x=1}^{s} \hat{p}(x)\log p(x;\theta) \quad \text{(where } \hat{p} \text{ is the empirical distribution)} \\
&= \operatorname*{argmin}_\theta -\sum_{x=1}^{s} \hat{p}(x)\log p(x;\theta) \quad \text{(the popular "cross-entropy" loss from machine learning)} \\
&= \operatorname*{argmin}_\theta \Big(\sum_{x=1}^{s} \hat{p}(x)\log\frac{\hat{p}(x)}{p(x;\theta)} + H(\hat{p}) \\
&= \operatorname*{argmin}_\theta D(\hat{p}\|p_\theta)
\end{aligned}$$

which makes sense since $D(\hat{p}\|p_\theta) = 0$ iff $p_\theta = \hat{p}$, and $\hat{p}$ should be heading for p as $n \to \infty$.

e.g. $X_{1:n} \sim$ iid $N(\mu, 1)$ μ unknown

$$\begin{aligned}
\text{Here } \theta = \mu, \quad f_\mu(x) &= \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(X-\mu)^2} \\
\Rightarrow \hat{\mu} &= \operatorname*{argmax}_{\tilde{\mu}} \prod_{k=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(X_k-\tilde{\mu})^2} \\
&= \operatorname*{argmax}_{\tilde{\mu}} e^{-\frac{1}{2}\sum_1^n (X_k-\tilde{\mu})^2} \\
&= \operatorname*{argmin}_{\tilde{\mu}} \sum_{1}^{n} (x_k-\tilde{\mu})^2 = \bar{X}
\end{aligned}$$

e.g. $X_{1:n} \sim$ iid $f_\lambda(x) = h(x)\frac{1}{Z_\lambda}\exp\{\sum_{l=1}^{c}\lambda_l T_l(x)\}$ ("exponential family with natural parameters"), where $\theta = \lambda = \vec{\lambda}$.

$$\begin{aligned}\hat{\lambda} &= \underset{\tilde{\lambda}}{\operatorname{argmax}} \prod_1^n f_{\tilde{\lambda}}(x_k) \\ &= \underset{\tilde{\lambda}}{\operatorname{argmin}} -\frac{1}{n}\sum_1^n \log f_{\tilde{\lambda}}(x_k) \\ &= \underset{\tilde{\lambda}}{\operatorname{argmin}} \left(\log Z_{\tilde{\lambda}} - \frac{1}{n}\sum_{k=1}^n \tilde{\lambda}\cdot T(x_k) \right)\end{aligned}$$

Recall that $\log Z_{\tilde{\lambda}} - \frac{1}{n}\sum_{k=1}^n \tilde{\lambda}\cdot T(x_k)$ is convex, and that $\dfrac{\partial}{\partial\tilde{\lambda}} \log Z_{\tilde{\lambda}} = \mathbb{E}_{\tilde{\lambda}}[T_l(X)]$

$$\Rightarrow \log Z_{\tilde{\lambda}} - \frac{1}{n}\sum_{k=1}^n \tilde{\lambda}\cdot T(x_k) \text{ is minimized when } \frac{\partial}{\partial\tilde{\lambda}_l}(\log Z_{\tilde{\lambda}} - \frac{1}{n}\sum_{k=1}^n \tilde{\lambda}\cdot T(x_k)) = 0 \ \forall l = 1,\ldots,c$$

$$\Rightarrow \frac{\partial}{\partial\tilde{\lambda}} \log Z_{\tilde{\lambda}} - \frac{\partial}{\partial\tilde{\lambda}_l}\frac{1}{n}\sum_1^n \tilde{\lambda}\cdot T(X_k) = 0 \ \forall l = 1,\ldots,c$$

$$\Rightarrow \mathbb{E}_{\tilde{\lambda}}[T_l(X)] = \frac{1}{n}\sum_1^n T_l(X_k) = \bar{T_l} \ \forall l = 1,\ldots,c$$

The set of equations, $\mathbb{E}_{\tilde{\lambda}}[T_l(X)] = \overline{T_l}$ $l = 1:c$, are called the "likelihood equations," and the solution, $\hat{\lambda}$, is called the maximum-likelihood estimator.

Theory

(1) When MLE's exist, they are typically consistent (b.c. $D(p_\theta \| p_{\hat{\theta}}) \Leftrightarrow p_{\hat{\theta}} = p_\theta$)

(2) When MLE's exists, they are typically asymptotically optimal, i.e. they achieve the minimum MSE as $n \to \infty$ (a property called "efficiency")

Failure Modes

(1) The model, p_θ, is wrong. For example, the true density might be of the form $f_\theta(x) = \frac{1}{2}\theta^2 x e^{-\theta x}$ but we use $g_\theta(x) = \theta e^{-\theta x}$. Maximum likelihood estimation (MLE) is generally *not* robust to model misspecification.

(2) Non-identifiability: Suppose that $\tilde{\theta} \neq \theta$, yet nevertheless $p_{\tilde{\theta}} = p_\theta$. Then θ is not identifiable (samples from p_θ and samples from $p_{\tilde{\theta}}$ are indistinguishable).

e.g. $f_\lambda(x) = \dfrac{1}{Z_\lambda}\exp\{\lambda_1 T_1(x) + \lambda_2 T_2(x)\}$

where $\alpha T_1(x) + \beta T_2(x) = \gamma$ for all x and some α, β, and γ. Then since $T_2(x) = \dfrac{\gamma}{\beta} - \dfrac{\alpha}{\beta}T_1(x)$

$$\begin{aligned}f_\lambda(x) &= \frac{1}{e^{-\frac{ac\gamma}{\beta}\lambda_2} Z_\lambda}\exp\{(\lambda_1 - \frac{\alpha}{\beta}\lambda_2)T_1(x)\} \\ &= \frac{1}{\tilde{Z}_\delta}\delta T_1(x)\}\end{aligned}$$

and we can only estimate $\delta \doteq \lambda_1 - \frac{\alpha}{\beta}\lambda_2$, and not λ_1 and λ_2.

(Even if $\alpha T_1(x) + \beta T_2(x)$ is only approximately equal to a constant γ, the likelihood $\prod_1^n f_\lambda(x_k)$ will nevertheless be very flat along the lines in $\mathbb{R}^2$ defined by $\lambda_1 - \frac{\alpha}{\beta}\lambda_2 = c$ for any constant c.)

(3) Nonparametrics: MLE typically fails for nonparametric estimation. For example, in nonparametric density estimation, the maximum of the likelihood

$$\underset{f}{\text{argmax}} = \prod_1^n f(X_k)$$

does not even exist. At least not as a density. The problem is that the maximum of the likelihood over all possible densities is infinite: for suppose that $X_1, X_2 \sim \ iid\ f$ and consider the kernel estimator

$$\hat{f}_\sigma = 0.5\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(x-X_1)^2}{2\sigma^2}} + 0.5\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(x-X_2)^2}{2\sigma^2}}$$

which places a kernel with bandwidth σ at each sample. The likelihood given the data, X_1 and X_2, for the estimator $\hat{f}_\sigma$ is

$$L = \hat{f}_\sigma(X_1)\hat{f}_\sigma(X_2) = \left(0.5\frac{1}{\sqrt{2\pi\sigma^2}} + 0.5\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(X_1-X_2)^2}{2\sigma^2}}\right)\left(0.5\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(X_2-X_1)^2}{\sigma^2}} + 0.5\frac{1}{\sqrt{2\pi\sigma^2}}\right)$$

which can be made arbitrarily large by making σ arbitrarily small. Hence no density achieves the maximum of the likelihood.

II Some statistical theory

B Classification

1. The classification problem
2. Definitions and terminology
3. Bayes optimal classification
4. Likelihood ratios and Neyman-Pearson optimal classification

1 The classification problem

Observe: $X \in \mathbb{R}^d$ image patch, diagnostic test results, $\cdots$

Classify: $Y \in \{1, 2, \cdots, s\}$ object type, diagnosis, $\cdots$

Notation

$\pi_c \quad c = 1, 2, \ldots, s$ class probabilities, $\mathbb{P}(Y = c) = \pi_c$

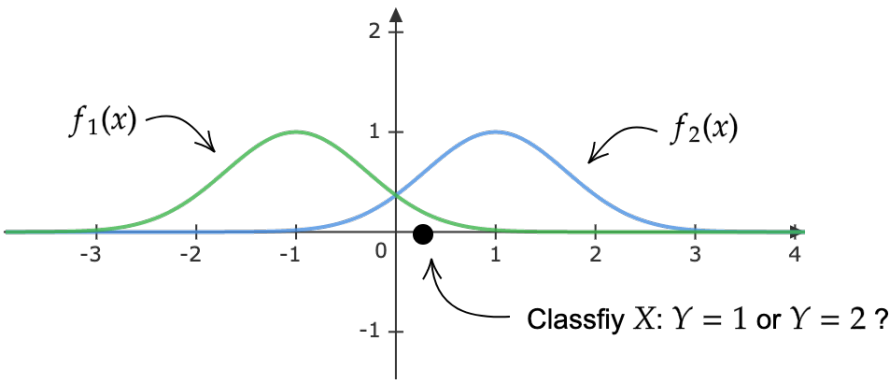
$P_c(A) = \mathbb{P}(X \in A | Y = c) \quad \forall A$ class-conditional probability distribution on X

$f_c(x) \quad x \in \mathbb{R}^d$ class-conditional density or mass function (pdf or pmf)

$P_X(A) = \sum_{c=1}^{s} \mathbb{P}_c(A)\pi_c \quad \forall A$ marginal probability distribution on X

$f_X(x) = \sum_{c=1}^{s} f_c(x)\pi_c \quad \forall x \in \mathbb{R}^d$ marginal probability density or mass function on X

e.g. $X \in \mathbb{R}$, $Y \in \{1, 2\}$, $f_1 = N(-1, 1)$, $f_2 = N(1, 1)$

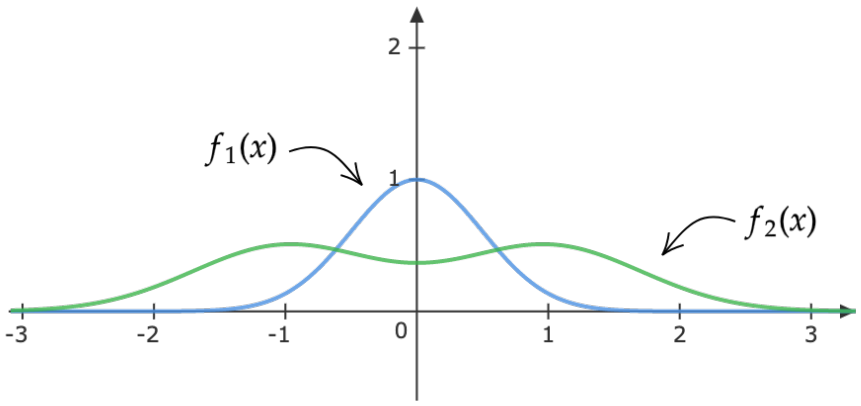


e.g. X =CA125, $Y = 2$ if ovarian cancer (or $X = PSA$, $Y = 2$ if prostate cancer)

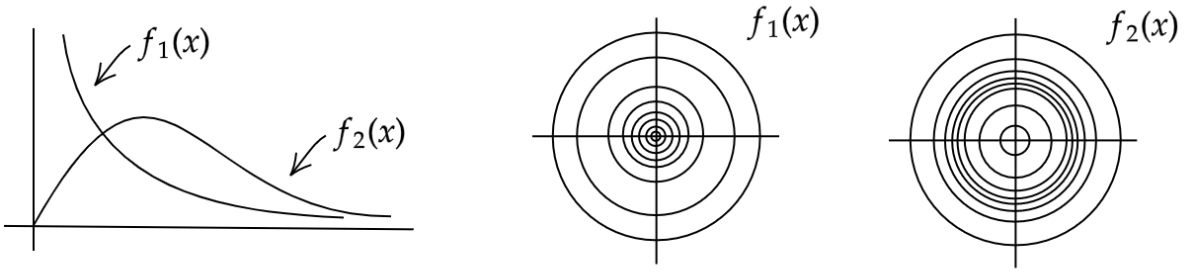
e.g. $f_1 = N(-1, 1) \times N(0, 1) \times \cdots \times N(0, 1), f_2 = N(1, 1) \times N(0, 1) \times \cdots \times N(0, 1)$. Only the first fea-

ture is relevant. But all of the other (irrelevant) features can make it very difficult to learn a good classifier.

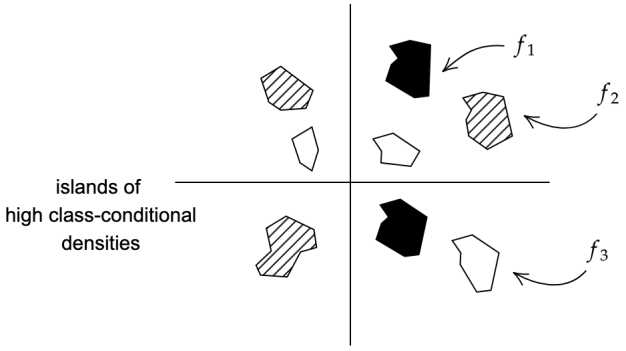
e.g. $X \in \mathbb{R}, Y \in \{1, 2\}, f_1 = N(0, 1/2), f_2 = \frac{1}{2}(N(-1, 1) + N(1, 1))$



e.g. $X \in \mathbb{R}^2, Y \in \{1, 2\}, f_1(r, \theta) = \frac{1}{2\pi}e^{-r}, f_2(r, \theta) = \frac{1}{2\pi}\frac{1}{24}r^4e^{-r}$



e.g. $X \in \mathbb{R}^2$, $Y = \{1, 2, 3\}$



2 Definitions

Classifiers

$$h : \mathbb{R}^d \to \{1, 2, \cdots, s\} \qquad h(x) = \text{ classification assigned to } x$$

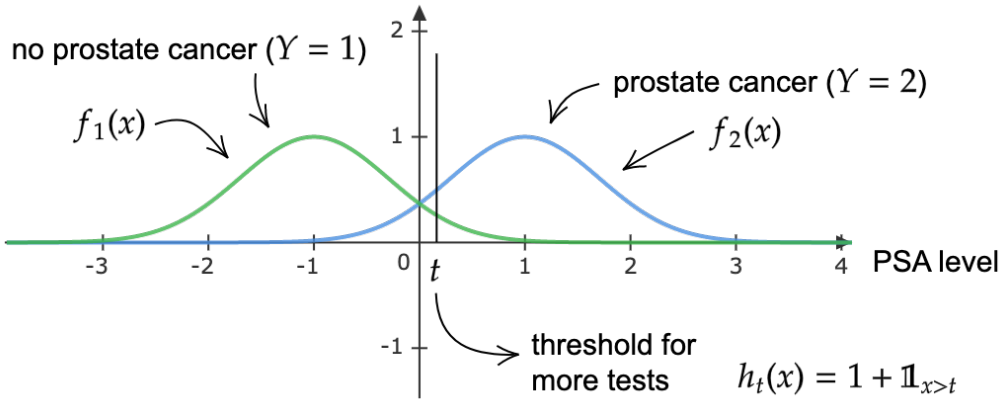
Performance of h

In general, based on "confusion matrix" $\mathbb{P}(h(X) = i | Y = j)$ $i, j \in \{1, 2, \cdots, s\}$

For $c = 2$, usually think of $Y = 2$ as "signal" or "positive result" and $Y = 1$ as "background," "null hypothesis," or "negative result," and then

$$\mathbb{P}(h(X) = 2 | Y = 2) \quad \begin{cases} \text{prob of detection} \\ \text{detection rate (DR)} \\ \text{sensitivity} \\ \text{power} \\ \text{1-(type II error rate)} \end{cases}$$

$$\mathbb{P}(h(X) = 2 | Y = 1) \quad \begin{cases} \text{prob of false alarm} \\ \text{false alarm rate (FAR)} \\ \text{1-specificity} \\ \text{size (as in } \alpha = 0.05) \\ \text{type I error rate} \end{cases}$$



$$\text{DR}_t = \int_t^\infty f_2(x)dx$$
$$\text{FAR}_t = \int_t^\infty f_1(x)dx$$

For a "screening test", willing to accept a relatively high FAR in order to have higher DR.

There are many perspectives on optimality of classifiers. Two prominent points of view are Bayes optimality and Neyman-Pearson optimality.

3 Bayes optimal classification

Minimize the classification error rate: Choose h to minimize $\mathbb{P}(h(X) \neq Y)$.

Thm ("Bayes Classification Rule" aka "maximum a posteriori" or "MAP" classifier)

Define $h^*(x) = \underset{c}{\operatorname{argmax}}\, \mathbb{P}(Y = c|X = x)$, then

$$\mathbb{P}(h^*(X) \neq Y) \leq \mathbb{P}(h(X) \neq Y) \;\; \forall h : \mathbb{R}^d \to \{1, \cdots, s\}$$

Pf (assume pdf for convenience)

Let $f_X(x) = \sum_c \pi_c f_c(x) =$ Marginal (unconditioned) density on X

Show, equivalently, $\mathbb{P}(Y = h^*(X)) \geq \mathbb{P}(Y = h(X))$:

$$\begin{aligned}
\mathbb{P}(Y = h^*(X)) &= \int_x \mathbb{P}(Y = h^*(X)|X = x) f_X(x) dx \\
&= \int_x \max_c \mathbb{P}(Y = c|X = x) f_X(x) dx \\
&\geq \int_x \mathbb{P}(Y = h(X)|X = x) f_X(x) dx \\
&= \mathbb{P}(Y = h(X))
\end{aligned}$$

4 Likelihood ratios and Neyman-Pearson optimal classification

For the 2 class problem ($Y \in \{1, 2\}$): maximize DR for any given FAR and/or minimize FAR for any given DR:

Thm (Neyman-Pearson lemma) For any $t \in (0, \infty)$, define

$$h_t(x) = \begin{cases} 1, & \text{if } \dfrac{f_2(x)}{f_1(x)} < t \\ 2, & \text{if } \dfrac{f_2(x)}{f_1(x)} > t \end{cases} \qquad (f_c \text{ could be pdf or pmf})$$

and let $h(x)$ be any classifier with

$$\text{FAR}_h \triangleq \mathbb{P}(h(X) = 2|Y = 1) \overset{(<)}{\leq} \mathbb{P}(h_t(X) = 2|Y = 1) \triangleq \text{FAR}_{h_t}$$

Then,

$$\text{DR}_h \triangleq \mathbb{P}(h(X) = 2|Y = 2) \overset{(<)}{\leq} \mathbb{P}(h_t(X) = 2|Y = 2) \triangleq \text{DR}_{h_t}$$

Remark: Ties ($f_2(x) = f_1(x)$) can be broken systematically or randomly.

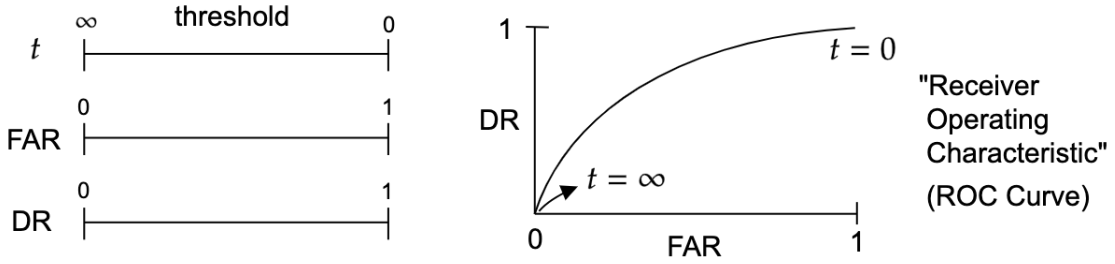
In words:

h_t has best possible DR among all classifiers with same (or better) FAR

or you can invoke the contrapositive:

h_t has best possible FAR among all classifiers with same (or better) DR

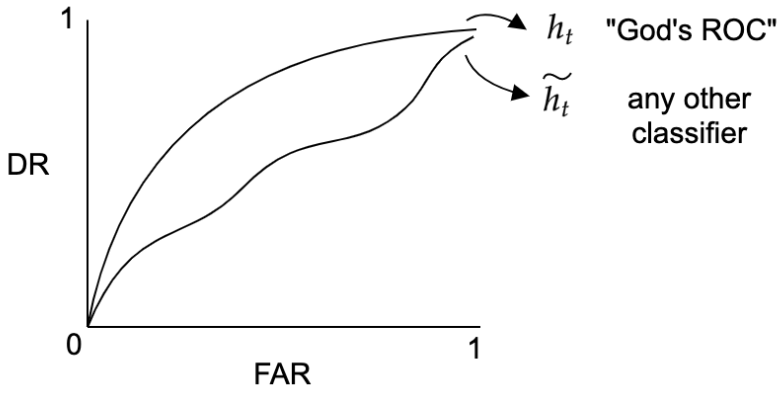
Notice that h_t is really a collection of classifiers: one for each t:



Suppose $\tilde{h}_t$ is another classifier. They are often of form

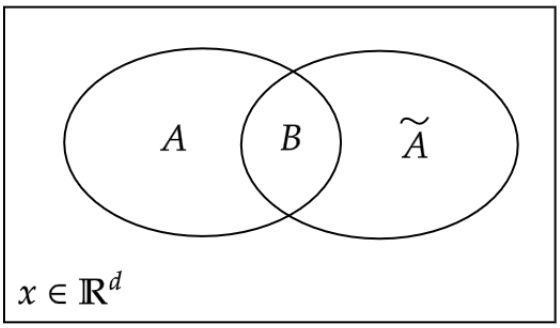
$$\tilde{h}_t(x) = \begin{cases} 1, & g(x) < t \\ 2, & g(x) > t \end{cases} \text{ for some } g$$

in which case $\tilde{h}_t$ also defines a collection of classifiers (one for each $t \geq 0$), and hence another ROC curve:



Proof of N-P Lemma $\begin{cases} h_t(x) & \text{N-P} \\ h(x) & \text{any other} \end{cases}$

Fix $t \in (0, \infty)$ and assume f_1, f_2 pdf's (same proof for pmf's but $\int_x \to \sum_x$)



$A \cup B: \; h(x) = 2$

$\widetilde{A} \cup B: \; h_t(x) = 2$

(Informally: h says "2" and h_t says "1" on A; h and h_t both say "2" on B; h says "1" and h_t says "2" on $\tilde{A}$.)

Observe $\quad A \subseteq \{x : \dfrac{f_2(x)}{f_1(x)} \leq t\} = \{x : f_2(x) \leq t f_1(x)\} \quad (+)$

$\qquad\quad \tilde{A} \subseteq \{x : \dfrac{f_2(x)}{f_1(x)} \geq t\} = \{x : f_2(x) \geq t f_1(x)\} \quad (*)$

Suppose $\text{FAR}_h \triangleq P_1(A \cup B) \overset{(<)}{\leq} P_1(\tilde{A} \cup B) \triangleq \text{FAR}_{h_t}$

Then,

$$\begin{aligned} P_1(A) &\overset{(<)}{\leq} P_1(\tilde{A}) \\ \Rightarrow t P_1(A) &\overset{(<)}{\leq} t P_1(\tilde{A}) \\ \Rightarrow \int_A t f_1(x) dx &\overset{(<)}{\leq} \int_{\tilde{A}} t f_1(x) dx \\ \underset{\text{by } (+)}{\geq} \Big\downarrow & \qquad\qquad \Big\downarrow \underset{\text{by } (*)}{\leq} \\ \Rightarrow \int_A f_2(x) dx &\overset{(<)}{\leq} \int_{\tilde{A}} f_2(x) dx \\ \Rightarrow P_2(A) &\overset{(<)}{\leq} P_2(\tilde{A}) \\ \text{i.e. } \text{DR}_h \triangleq P_2(A \cup B) &\overset{(<)}{\leq} P_2(\tilde{A} \cup B) \triangleq \text{DR}_{h_t} \end{aligned}$$

Remarks on N-P

1. $N-P$ Classifiers makes no reference to $\pi_1 = \mathbb{P}(Y=1)$ or $\pi_2 = \mathbb{P}(Y=2)$
2. Recall Bayes' Classifier: if 2 classes

$$\begin{aligned} h^*(x) &= \underset{c\in\{1,2\}}{\operatorname{argmax}} \mathbb{P}(Y=c|X=x) \\ &= \underset{c\in\{1,2\}}{\operatorname{argmax}} \frac{\pi_c f_c(x)}{\pi_1 f_1(x) + \pi_2 f_2(x)} \\ &= \underset{c\in\{1,2\}}{\operatorname{argmax}} \pi_c f_c(x) \\ &= \begin{cases} 1 & \text{if } \pi_1 f_1(x) > \pi_2 f_2(x) \\ 2 & \text{if } \pi_1 f_1(x) < \pi_2 f_2(x) \end{cases} \\ &= \begin{cases} 1 & \text{if } f_2(x)/f_1(x) < \pi_1/\pi_2 \\ 2 & \text{if } f_2(x)/f_1(x) > \pi_1/\pi_2 \end{cases} \\ &\Leftrightarrow \text{N-P at } t = \pi_1/\pi_2 \end{aligned}$$

   Hence, $t = \pi_1/\pi_2$ minimizes overall error rate, since we proved this of the Bayes Classfifier!

3. Weighted Bayes decision rule

$$h^*(x) = \underset{c}{\operatorname{argmax}} \mathbb{P}(Y=c|X=x) = \begin{cases} 1 & \text{if } f_2(x)/f_1(x) < \dfrac{w_1\pi_1}{w_2\pi_2} \\ 2 & \text{if } f_2(x)/f_1(x) > \dfrac{w_1\pi_1}{w_2\pi_2} \end{cases}$$

$$= \text{N-P at } t = \frac{w_1\pi_1}{w_2\pi_2}$$

Aside In machine learning, it is popular to use "precision" and "recall" (instead of FAR and DR)

$$\begin{aligned} \text{Recall} &\triangleq \mathbb{P}(h(X)=2|Y=2) \ (=\text{DR}) \\ \text{Precision} &\triangleq \mathbb{P}(Y=2|h(X)=2) \end{aligned}$$

So precision is fundamentally Bayesian in concept (since it requires a prior probability on Y - i.e. π_1 and π_2):

$$\text{Precision} = \frac{\pi_2 \cdot \text{DR}}{\pi_2 \cdot \text{DR} + \pi_1 \cdot \text{FAR}}$$

II Some statistical theory

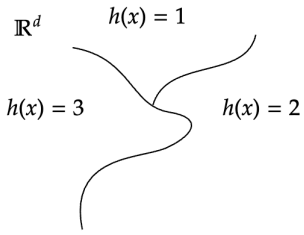
C Building classifiers

1. Generative construction
2. Discriminative construction
3. Algorithmic construction

How do we actually construct $h : \mathbb{R}^d \to \{1, 2, \cdots, s\}$? Usually we are just given sample data, i.e. observations of pairs of $X \in \mathbb{R}^d$ and $Y \in \{1, 2, \cdots, s\}$:

$$(X_1, Y_1), \cdots, (X_n, Y_n)$$

(rather than $f_1(x), \cdots, f_s(x), \pi_1, \cdots, \pi_s$) The problem of classification comes down to computing decision boundaries:



Broadly, there are three types of approaches:

Generative
e.g. nonparametric, naive Bayes, quadratic discriminant analysis

Discriminative
e.g. softmax, logistic regression, k nearest neighbor

Algorithmic
e.g. support vector machines, neural nets

1 Generative Construction

e.g. **"non-parametric"**

One (ambitious!) approach would be to attempt to construct the Bayes-optimal classifier, estimating the class probabilities and the class-conditional densities non-parametrically.

$$h^*(x) = \operatorname*{argmax}_c \mathbb{P}(Y = c | X = x) = \operatorname*{argmax}_c \pi_c f_c(x) \approx \operatorname*{argmax}_c \widehat{\pi}_c \widehat{f}_c(x)$$

where

$$\widehat{\pi}_c = \frac{\#\{i : Y_i = c\}}{n} \to \pi_c \quad \text{fast and efficient; not a problem}$$

$\widehat{f}_c(x)$, for example, the kernel estimator, but this can be trouble if $x \in \mathbb{R}^d, d >> 1$ ("curse of dimensionality")

There are many approaches designed to mitigate these difficulties:

e.g. **"naive Bayes"** $x \in \mathbb{R}^d \quad x = (x_1, \cdots, x_d)$, with the model

$$f_c(x) = f_c(x_1, \cdots, x_d) = \prod_{k=1}^{d} f_{c,k}(x_k)$$

i.e. we *assume* that the features are independent given the classfication, c.

Then

$$\widehat{f}_c(x) = \prod_{k=1}^{d} \widehat{f}_{c,k}(x_k)$$

and we can break up the data into class-specific feature vectors:

$$X(c,i) \doteq (X_1(c,i), \cdots, X_d(c,i)) \in \mathbb{R}^d \quad c = 1:s, i = 1, 2, \cdots, n_c$$

where n_c is the number of samples with $Y = c$. These are then used to estimate corresponding feature distributions, one for each class and each dimension:

$$\widehat{f}_{c,k}(x) = \widehat{f}_{c,k}(x; X_k(c,1), \cdots, X_k(c,n_c))$$

The point being that we are estimating $d \times s$ 1-dimensional densities rather than s d-dimensional densities, which is a whole lot easier.

This is a big win on variance, but at the expense of bias—unless, of course, the features really are (at least nearly) independent given the classification.

e.g. **"quadratic discriminant analysis"** Model: $f_c(x) \sim N(\vec{\mu}_c, \Sigma_c)$, where $\vec{\mu}_c$ has d parameters and Σ_c (the covariance matrix, which is symmetric) has $(d^2 + d)/2$ parameters:

$$f_c(x; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_c|}} \exp\left(-\frac{1}{2}(x - \vec{\mu}_c)^T \Sigma_c^{-1} (x - \vec{\mu}_c)\right)$$

MLE would generally be OK, although we would want the number of samples for each category to be many times greater than $(d^2 + 3d)/2$, the number of parameters for each category.

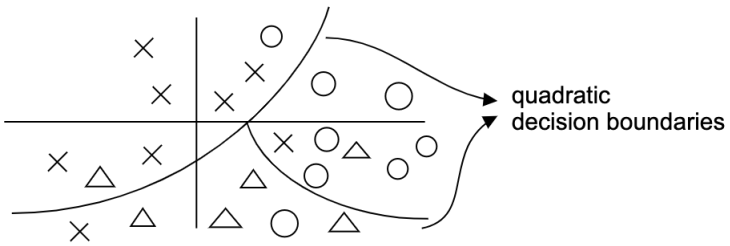
Once the model is estimated, we can invoke Neyman-Pearson theory or take the Bayesian approach, both of which lead to decision boundaries that are defined by thresholding likelihood ratios:

$$\frac{f_i(x)}{f_j(x)} \geq t \quad \text{(Neyman Pearson)} \quad \text{or} \quad \frac{f_i(x)}{f_j(x)} \geq \frac{\pi_j}{\pi_i} \quad \text{(Bayesian)}$$

Either way, for any $i, j \in \{1, \ldots, s\}$ and threshold α_{ij}:

$$\begin{aligned}
\frac{f_i(x)}{f_j(x)} \geq \alpha_{ij} &\Leftrightarrow \frac{\frac{1}{\sqrt{(2\pi)^d |\Sigma_i|}} \exp\left(-\frac{1}{2}(x - \vec{\mu}_i)^T \Sigma_i^{-1} (x - \vec{\mu}_i)\right)}{\frac{1}{\sqrt{(2\pi)^d |\Sigma_j|}} \exp\left(-\frac{1}{2}(x - \vec{\mu}_j)^T \Sigma_j^{-1} (x - \vec{\mu}_j)\right)} \geq \alpha_{ij} \\
&\Leftrightarrow -(x - \vec{\mu}_i)^T \Sigma_i^{-1} (x - \vec{\mu}_i) + (x - \vec{\mu}_j)^T \Sigma_j^{-1} (x - \vec{\mu}_j) + \log\left(\frac{\sqrt{(2\pi)^d |\Sigma_j|}}{\sqrt{(2\pi)^d |\Sigma_i|}}\right) \geq \log \alpha_{ij} \\
&\Leftrightarrow (x - \vec{\mu}_j)^T \Sigma_j^{-1} (x - \vec{\mu}_j) - (x - \vec{\mu}_i)^T \Sigma_i^{-1} (x - \vec{\mu}_i) \geq \tilde{\alpha}_{ij} \qquad (1)
\end{aligned}$$

for some $\tilde{\alpha}_{ij}$. The ties, which occur when the left side of (1) is set equal to $\tilde{\alpha}_{ij}$, define the "decision boundaries," which in this case are defined by a set of quadratic equations, one for every pair $i, j \in \{1 \ldots, s\}$ of categories.



e.g. **"linear discriminant analysis"** We can drastically reduce the number of parameters by assuming that the covariance function is shared across classes: $f_c(x) \sim N(\vec{\mu}_c, \Sigma)$ (so Σ is independent of c). The quadratic terms in equation (1) drop out, resulting in a linear equation for the decision boundaries which are therefore planar, i.e. linear:

$$(\vec{\mu_j} - \vec{\mu_i})^T \Sigma^{-1} x = \tilde{\alpha}_{i,j} \quad \forall\ i, j$$

2 Discriminative construction

The goal is the same: find a good estimator for

$$h^*(x) = \operatorname*{argmax}_c \mathbb{P}(Y = c | X = x)$$

But why bother with π_c and f_c to get $\mathbb{P}(Y = c|X = x)$? Do we really need a fully generative model? Why not, instead, go right after the functions

$$r_c(x) \triangleq \mathbb{P}(Y = c|X = x) \quad c = 1 : s$$

After all, this is familiar territory: we can rewrite $r_c(x)$ as $r_c(x) = \mathbb{E}[\mathbb{1}_{Y=c}|x]$, and now think of $\mathbb{1}_{Y=c}$ as the "dependent variable" and x is the "independent variable," which brings us to a variation on ordinary regression.

In fact, it doesn't even matter whether x is random or not, we will simply try to estimate $r_c(x)$ for every x. (Though there is one constraint that we need to keep in mind. Namely that $\sum_c r_c(x) = 1$, since the sum of the probabilities of the categories, given that $X = x$, must be one.)

We can do this parametrically or non-parametrically. We will look at examples of both.

e.g. **"logistic regression"** This is the prototypical parametric approach, with $s = 2$. Since, in this case, $r_1(x) = 1 - r_2(x)$, it is enough to specify a model for $r_2(x)$. The logistic model depends on a scalar parameter $\alpha \in \mathbb{R}$ and a vector parameter $\beta \in \mathbb{R}^n$, so $r_2(x) = r_2(x; \alpha, \beta)$:

$$\log \frac{r_2(x)}{1 - r_2(x)} = \alpha + \beta \cdot x \quad x \in \mathbb{R}$$

In other words, $\log \frac{r_2}{r_1}$ (the "log-odds ratio") is modeled as a simple linear regression. And since $\frac{a}{1-a} = b \Leftrightarrow a = \frac{b}{1+b}$, the model can also be written more directly as

$$r_2(x) = \frac{e^{\alpha+\beta\cdot x}}{1 + e^{\alpha+\beta\cdot x}}$$

Generally, we would use MLE to estimate r_2, given the data $\{(x_k, y_k)\}_{k=1:n}$, $x_k \in \mathbb{R}^d, y_k \in \{1, 2\}$. Notice that X_k could be random or not, which is a general feature of discriminative approaches. The likelihood is the product of the likelihoods of the individual observations, under the assumption that the labels (the Y's) are conditionally independent given the (deterministic or random) x's:

$$L(\alpha, \beta) = \prod_{k=1}^{n} r_2(x_k)^{y_k - 1}(1 - r_2(x_k))^{2 - y_k} \qquad \text{(independent but not identically binary random variables)}$$

Alternatively, since $1 - r_2(x) = r_1(x)$,

$$L(\alpha, \beta) = \prod_{k=1}^{n} r_{y_k}(x_k)$$

The MLE is then the argmax of L, with respect to α and β ($r_2(x) = r_2(x; \alpha, \beta)$).

As for the classification rule, fix a threshold t and, as usual, declare '2' if $\dfrac{r_2(x)}{r_1(x)} > t$, which is the same as $\alpha + \beta \cdot x = \tau$ for some tau, i.e. a linear decision boundary.

e.g. **"softmax"** This is the natural extension of logistic regression to $s > 2$.

Model: $\log \dfrac{r_k(x)}{r_1(x)} = \alpha_k + \beta_k \cdot x \quad k = 2 : s$

$$\Rightarrow \frac{r_k(x)}{1 - \sum_{l=2}^{s} r_l(x)} = e^{\alpha + \beta_k \cdot x} \quad k = 2 : s \qquad (*)$$

$$\Rightarrow \frac{\sum_{l=2}^{s} r_l(x)}{1 - \sum_{l=2}^{s} r_l(x)} = \sum_{k=2}^{s} e^{\alpha + \beta_k \cdot x} \qquad (\text{since } \textstyle\sum_{l=2}^{s} r_l = \sum_{k=2}^{s} r_k)$$

$$\Rightarrow 1 - \sum_{l=2}^{s} r_l(x) = \frac{1}{1 + \sum_{2}^{s} e^{\alpha + \beta_k \cdot x}} \qquad (\tfrac{a}{1-a} = b \Leftrightarrow 1 - a = \tfrac{1}{1+b}) \qquad (+)$$

$$\Rightarrow r_k(x) = \frac{e^{\alpha + \beta_k \cdot x}}{1 + \sum_{2}^{s} e^{\alpha + \beta_k \cdot x}} \quad k = 2 : s \qquad (\text{use } (+) \text{ in } (*))$$

As for inference ("training"), this comes down to likelihood, as in logistic regression, but with more parameters: set

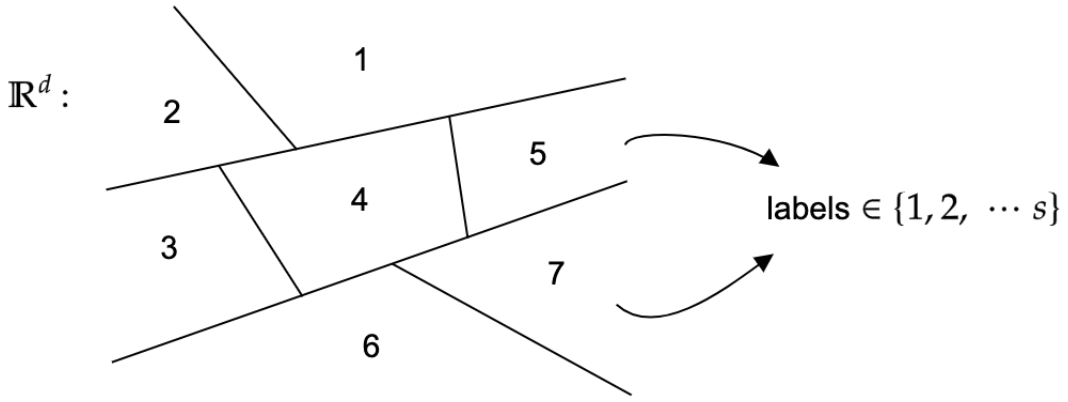
$$L(\{\alpha_c\}_{c=2}^{s}, \{\beta_c\}_{c=2}^{s}; \{y_k, x_k\}_{k=1}^{n}) = \prod_{k=1}^{n} r_{y_k}(x_k)$$

and maximize over the parameters. (This is equivalent to minimizing the "cross-entropy loss function" which is much used in machine learning.)

The decision boundaries come from comparing $\frac{r_k(x)}{r_l(x)}$ to a threshold:

$$\frac{r_k(x)}{r_l(x)} = t \Leftrightarrow \alpha_k + \beta_k \cdot x = t\alpha_l + t\beta_l \cdot x$$

which again defines linear decision boundaries.



Embedding Recall that the model is

$$\log \frac{r_c(x)}{r_1(x)} = \alpha_c + \beta_c \cdot x$$

which is a linear regression model, meaning *linear in the parameters*. An important observation is that this key property is preserved even if we include *nonlinear* features, e.g.

$$\log \frac{r_c(x)}{r_1(x)} = \alpha_c + \beta_c \cdot x + \sum_{i=1}^{d} \sum_{j=1}^{i} \beta_{ij}^c x_i x_j$$

is still linear in the parameters.

More generally, if ϕ is any mapping from $\mathbb{R}^d$ to $\mathbb{R}^{d'}$, $\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$ (an "embedding") then we can just as easily use $\mathbb{R}^{d'}$ as the feature space:

$$\begin{aligned} \log \frac{r_c(x)}{r_1(x)} &= \alpha_c + \beta_c \cdot \phi(x) \text{ (where } \beta_c \in \mathbb{R}^{d'}) \\ &= \alpha_c + \sum_{k=1}^{d'} \beta_k^c \cdot \phi_k(x) \end{aligned}$$

Notice that the resulting decision surfaces will be *linear* in $\mathbb{R}^{d'}$ but *nonlinear* in the original feature space, $\mathbb{R}^d$!

If, for example, $d = 3$, we might replace $x = (x_1, x_2, x_3)$ with

$$\phi(x) = (x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3, x_1^2, x_2^2, x_3^2) \in \mathbb{R}^9$$

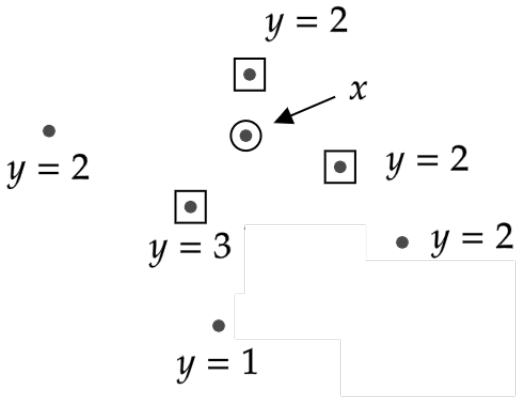
in which case $\beta_c = (\beta_1, \beta_2, \beta_3) \in \mathbb{R}^3$ is replaced by $\beta_c = (\beta_1, \beta_2, \ldots, \beta_9) \in \mathbb{R}^9$

Any functions of x can be used for the components, $\phi_1(x), \phi_2(x), \ldots, \phi_{d'}$ of $\phi(x) \in \mathbb{R}^{d'}$, e.g.

$$c^{x_1}, \cos(x_1x_2), x_1^3, x_1x_2x_3, \cdots$$

The choice of components, and hence features, is part of what is sometimes referred to as "feature engineering."

e.g. **"k nearest neighbor classifier"** This is the prototypical nonparametric classifier. For any $k \in \{1, 2, \cdots\}$ (which will be turn out to be the smoothing parameter) $r_c(x),\ c = 1 : s$ is the outcome of a vote by the k nearest data points to x:



say $k = 3,\ s = 3$

3 nearest nbrs (" □ ") vote:

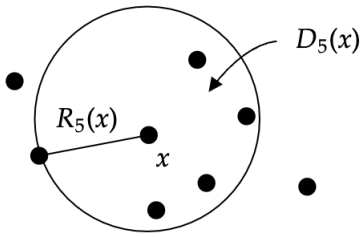
$$r_1(x) = \frac{0}{3}$$
$$r_2(x) = \frac{2}{3}$$
$$r_3(x) = \frac{1}{3}$$

The nearest neighbor has label $y = 2$, and hence the nearest neighbor classifier, at x, is $\hat{r}_1(x) = (0, 1, 0)$. Similarly, the three nearest neighbors include two 2s, and one 3, and hence $\hat{r}_3(x) = (0, 2/3, 1/3)$, and the six nearest neighbors have labels 1,2,2,2,2 and 3, and hence $\hat{r}_6(x) = (1/6, 4/6, 1/6)$.

Formally, define

$$R_k(x) = R_k(x; X_{1:n}) = \text{ distance to } k^{th} \text{ nearest nbr } = \inf\{\varepsilon > 0 : \#\{i : \|x_i - x\| \leq \varepsilon\} \geq k\}$$

and let $D_k(x)$ be the closed disk (ball) centered at x with radius $R_k(x)$:



then $\widehat{r}_c(x) = \widehat{r}_{c,k}(x) = \dfrac{\#\{i : X_i \in D_k(x) \text{ and } Y_i = c)\}}{\#\{i : X_i \in D_k(x)\}}$

To see why this makes sense (and outline a consistency proof):

Fix c and let $B_\varepsilon(x)$ be the closed ball of radius $\varepsilon \in \mathbb{R}^d$ centered at $x \in \mathbb{R}^d$

$$\frac{\#\{i : X_i \in B_\varepsilon(x) \text{ and } Y_i = c\}}{\#\{i : X_i \in B_\varepsilon(x)\}} \qquad \text{(i.e. } \widehat{r}_{c,k}(x) \text{ when } \varepsilon = R_k(x)) \qquad \text{(i)}$$

$$\xrightarrow{n\to\infty} \frac{\mathbb{P}(X \in B_\varepsilon(x) \text{ and } Y = c)}{\mathbb{P}(X \in B_\varepsilon(x))}$$

$$= \frac{\mathbb{P}(X \in B_\varepsilon(x)|Y = c)\pi_c}{\mathbb{P}(X \in B_\varepsilon(x))}$$

$$= \frac{\pi_c \int_{B_\varepsilon(x)} f_c(\tilde{x})d\tilde{x}}{\mathbb{P}(X \in B_\varepsilon(x))} \qquad \text{(to keep it simple, assume } f_c\text{'s are densities)}$$

$$= \frac{\frac{\pi_c}{|B_\varepsilon(x)|} \int_{B_\varepsilon(x)} f_c(\tilde{x})d\tilde{x}}{\frac{1}{|B_\varepsilon(x)|} \int_{B_\varepsilon(x)} f(\tilde{x})d\tilde{x}}$$

$$\xrightarrow{\varepsilon\to 0} \frac{f_c(x)\pi_c}{f(x)} = \mathbb{P}(Y = c|X = x) \qquad \text{(ii)}$$

Notice that $R_k(x) \to 0$ as $n \to \infty$ for fixed k. Now take $k = k_n \to \infty$ ("kill the variance", see (i)) but slowly enough that $\varepsilon = \varepsilon_n = R_k(x) \to 0$ ("kill the bias", see (ii)). Bias/Variance tradeoff: $\widehat{r}_c(x) = \widehat{r}_c(x; k_n) \to \mathbb{P}(Y = c|X = x)$ if

(i) $k_n \to \infty$ ("kill the variance"), and

(ii) $\dfrac{k_n}{n} \to 0$ ("kill the bias")

In general: k small $\Rightarrow$ high variance low bias
k large $\Rightarrow$ low variance high bias

Finally, here are some asymptotic results that we won't prove. Fix k, let $L_k =$ be the limiting ($n \to \infty$) classification error, and let L^* be the Bayes optimal classification error. Then

- $L^* \leq L_k \ \forall k$ (obviously)
- $L_{k+1} \leq L_k$
- $L_k \leq L^* \cdot \left(1 + \sqrt{\dfrac{2}{k}}\right)$
- $L_1 \leq 2L^*$

But keep in mind that these results are asymptotic, and not very relevant for a fixed, finite, data set.

II Some statistical theory

C Building classifiers

Support Vector Machine (SVM)

a. Algorithmic construction
b. Maximum margin classifier
c. Soft margin and the control of overfitting
d. Embedding and the kernel trick
e. Positive definite kernels

a. Algorithmic construction

If all we care about are decision surfaces, why bother estimating

$$r_c(x) = \mathbb{P}(Y = c|X = x)?$$

Consider, for example, the case $s = 2$, and suppose that we've estimated $r_2(x)$ (and hence also $r_1 = 1 - r_2$). Then we would presumably use these to make decisions based on the ratio $r_2(x)/r_1(x)$, as in

$$h(x) = \begin{cases} 2 & \text{if } \dfrac{r_2(x)}{r_1(x)} > 1 \\ 1 & \text{if } \dfrac{r_2(x)}{r_1(x)} < 1 \end{cases}$$
$$= 1 + \mathbb{1}_{r_2(x) > r_1(x)}$$

Instead of estimating $r_2(x)$, we could just estimate the decision boundary directly:

$$\{x : r_2(x) = r_1(x)\}$$

We will call this the "algorithmic approach", and we will focus on $s = 2$. There are many ways to generalize to $s > 2$.

A small change in notation will make the mathematics a lot cleaner: henceforth we will assume that $Y \in \{-1, 1\}$ instead of $\{1, 2\}$.

A big part of the SVM story is embedding, which we have already discussed in the context of softmax. We will call a training set consistent if whenever two examples have the same feature vector they also have the same label. Later we will show that there exists universal embedding functions such that any consistent training set is transformed into a linearly separable training set. This is part of what is known as the "kernel trick." Thus we will

(1) Choose a nonlinear embedding function $\phi : R^d \to R^{d'}$

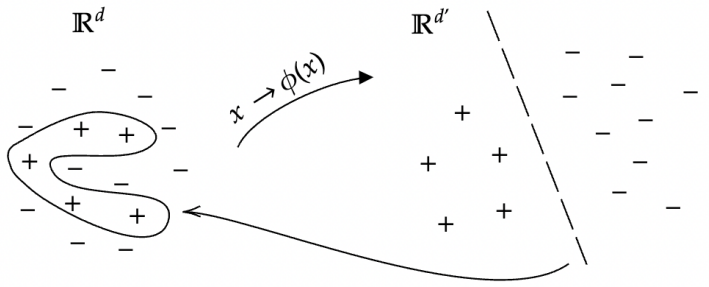


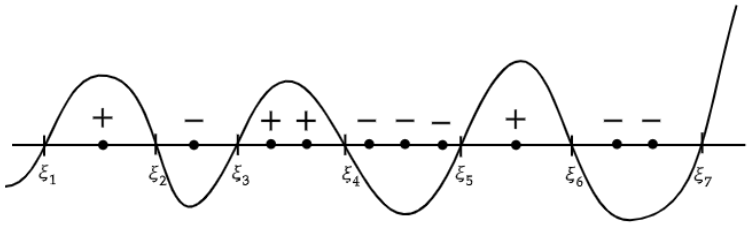
Figure 1: **Embedding.** A nonlinear embedding $\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$ can render a training set linearly separable in the embedding space. The linear decision surface, as viewed in the original space ($\mathbb{R}^d$), can be quite complicated.

(2) Fit a linear discriminant boundary in $\mathbb{R}^{d'}$ to the embedded data

$$(\phi(X_1), Y_1), \cdots, (\phi(X_n), Y_n)$$

(3) Return to the ambient feature space, $\mathbb{R}^d$, where the corresponding boundary will be nonlinear

As a simple example, consider $d = 1$:



$$h(x) = \text{sign } (\prod_{k=1}^{7}(x - \xi_k))$$
$$= \text{sign } (\alpha + \prod_{k=1}^{7} \beta_k x^k) = \text{sign } (\alpha + \beta \cdot \phi(x))$$

where $\phi(x) = (x, x^2, x^3, x^4, x^5, x^6, x^7) \in \mathbb{R}^7$ (or $\mathbb{R}^8$ if it includes $\alpha : \phi(x) = (x^0, x, x^2, \cdots, x^7) \in \mathbb{R}^8$). Notice that increasing or decreasing α shift the polynomial upwards or downwards, favoring, respectively, higher detection rates or lower false positive rates.

Obviously there is a danger of overfitting when d' is large. In general:
d' small: high bias, low variance
d' large: low bias, high variance

b. Maximum margin classifier

For now we will put aside the embedding problem and *assume that the data is already linearly separable.* There is still an issue of avoiding high variance: which of the infinity of linear boundaries should we select?

Consider Figure (2). Obviously, there are many to choose from, but a compelling case can be made for the linear classifier that maximizes the distance between the decision surface and the closest feature vector in the training data. This is called the maximum margin classifier.

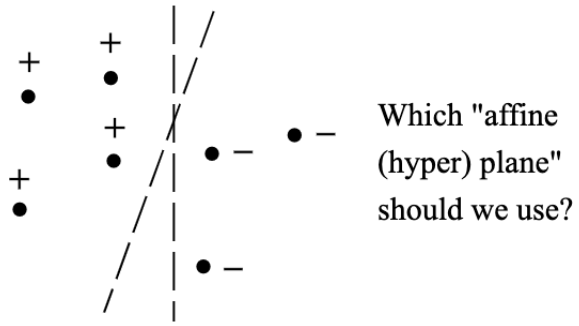


Figure 2: **Which linear classifier?**

Assume that $\{(X_k, Y_k)\}_{k=1:n}$ is linearly separable, and define $H(x) = \alpha + \beta \cdot x = \alpha + \sum_{i=1}^{d} \beta_i x_i$. Then $H(x)$ defines a classifier:

$$h(x) = \text{sign } H(x)$$

Since only the direction of $\beta \in \mathbb{R}^d$ matters, we can assume WLOG that $\|\beta\| = 1$. Notice that $H(x) = a$ defines parallel planes, one for every $a \in \mathbb{R}$, and since $\|\beta\| = 1$, the two planes $H(x) = a$ and $H(x) = a'$ are distance $|a - a'|$ apart (see Figure (3)).

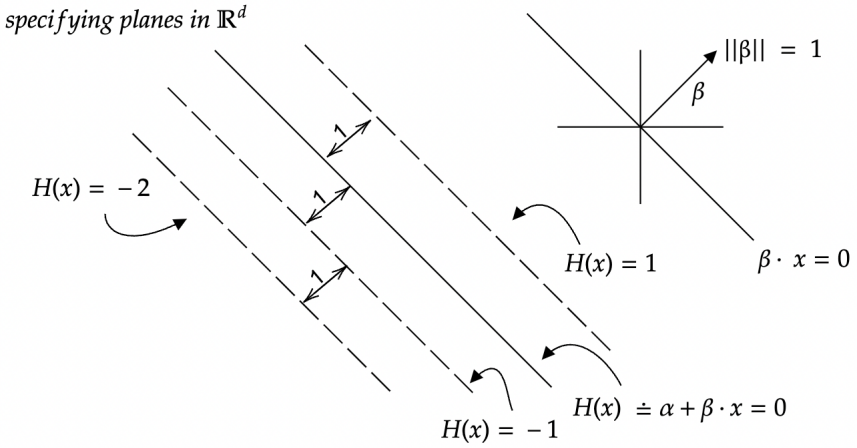


Figure 3: **Normalized** β**.** When $\|\beta\| = 1$, the magnitude of $H(x)$ is the distance from x to the plane $H(x) = 0$

The idea is to choose α, β so that

(i) $h(x) = \text{sign } H(x)$ perfectly separates the data

(ii) the minimum distance to $H(x) = 0$, over all $X_{1:n}$, is maximized (see Figure 4)

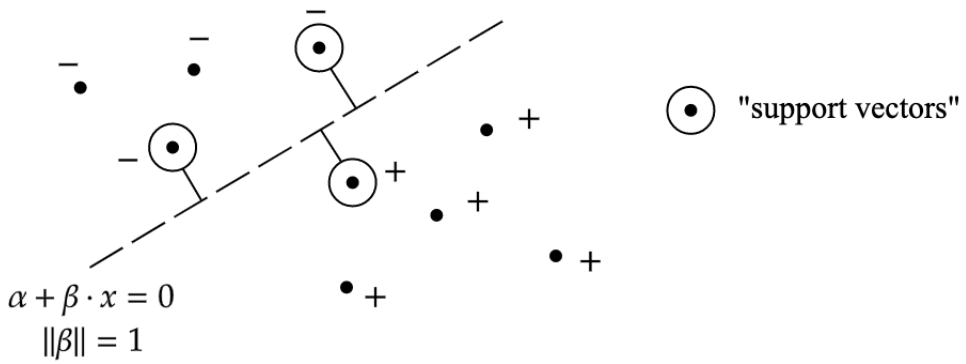


Figure 4: **Support Vectors.** Assume that the data is linearly separable. The support vector machine is the linear classifier that achieves the maximum distance, M, to the nearest feature vectors. The "support vectors" are the vectors that are at exactly distance M from the decision surface.

Conclude that the maximum-margin linear separator solves the following optimization problem:

$$(\widehat{\alpha}, \widehat{\beta}) = \underset{\alpha,\beta,\|\beta\|=1}{\operatorname{argmax}} M \qquad (*)$$
$$\text{subject to } Y_k(\alpha + \beta \cdot X_k) \geq M \quad \forall k = 1 : n$$

A *convex optimization problem* is one in which we minimize a convex function over a convex set. Typically, convex optimization is much easier than non-convex optimization. Unfortunately, $(*)$ is not a convex problem since the constraint $\|\beta\| = 1$ does not define a convex set. (Recall that a set is convex if the line connecting any two points in the set is completely contained within the set; a circle is not convex.) But $(*)$ can be converted to a convex problem, as follows. Divide the constraint

$$Y_k(\alpha + \beta \cdot X_k) \geq M$$

by M to get

$$Y_k\left(\frac{\alpha}{M} + \frac{\beta}{M} \cdot X_k\right) \geq 1$$

and now redefine α and β: $\dfrac{\alpha}{M} \to \alpha$, $\dfrac{\beta}{M} \to \beta$.

This leads to the equivalent, but *convex*, optimization problem

$$(\widehat{\alpha}, \widehat{\beta}) = \operatorname{argmin} \frac{1}{2}\sum_{i=1}^{d} \beta_i^2 \qquad (**)$$
$$\text{subject to } Y_k(\alpha + \beta \cdot X_k) \geq 1 \quad \forall k = 1 : n$$

The margin is now $1/\|\widehat{\beta}\|$. (We will forgo the details of the equivalence, $(*) \leftrightarrow (**)$.)

The win is that $(**)$ can be solved more or less directly. Better yet, we can introduce Lagrange multipliers and solve instead the "Lagrangian Dual." Again, we will leave out the details, which are standard fare in optimization theory. In brief:

(i) Fix $\lambda = (\lambda_1, \cdots, \lambda_n)$ subject to $\lambda_i \geq 0 \quad i = 1:n$

(ii) Find $\alpha = \alpha(\lambda)$, $\beta = \beta(\lambda)$ to minimize

$$(+) \quad \frac{1}{2}\sum_{k=1}^{d} \beta_k^2 + \sum_{i=1}^{n} \lambda_i (1 - Y_i(\alpha + \beta \cdot X_i))$$

(iii) maximize $(+)$ over λ, where $\alpha = \alpha(\lambda)$ and $\beta = \beta(\lambda)$

which leads to

Choose $(\widehat{\alpha}, \widehat{\beta}, \widehat{\lambda})$ to

$$\underset{\lambda \geq 0}{\text{maximize}} \ \underset{\alpha,\beta}{\text{minimum}} \ \tfrac{1}{2}\textstyle\sum_1^d \beta_k^2 + \sum_1^n \lambda_i (1 - Y_i(\alpha + \beta \cdot X_i)) \qquad (***)$$

If we write $\hat{\alpha}$ and $\hat{\beta}$ in terms of the positive constants $\widehat{\lambda}_1, \ldots, \widehat{\lambda}_n$, we find that

(i) $\widehat{\beta} = \sum_{i=1}^n \widehat{\lambda}_i Y_i X_i$

(ii) $\widehat{\lambda} \geq 0$, and $\widehat{\lambda}_i > 0 \Leftrightarrow X_i$ is a support vector

(iii) For any support vector X_i,

$$Y_i(\widehat{\alpha} + \widehat{\beta} \cdot X_i) = 1$$
$$\Rightarrow \ \widehat{\alpha} = Y_i - \widehat{\beta} \cdot X_i$$

(iv) And finally, since $h(x) = \text{sign}H(x) = \text{sign}(\widehat{\alpha} + \widehat{\beta} \cdot x)$,

$$h(x) = \text{sign}(\widehat{\alpha} + \sum_{i=1}^{n} \widehat{\lambda}_i Y_i X_i \cdot x)$$

Here are two important observations about the solution to $(***)$ that you should take note of:

(1) As a consequence of (ii), only the support vectors contribute to the final classifier. (Which also follows, more or less directly, from the definition of the maximum-margin classifier.)

(2) Define a "kernel function" $\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ by $\kappa(x, \tilde{x}) \triangleq x \cdot \tilde{x}$ (i.e. the inner product), in which case we can rewrite h as

$$h(x) = \text{sign}(\widehat{\alpha} + \sum_{i=1}^{n} \widehat{\lambda}_i Y_i \kappa(X_i, x))$$

In particular, the evaluation of h at a feature X_j, $h(X_j)$, for any $j \in \{1, 2, \ldots, n\}$, depends only on the collection of inner products, $\{\kappa(X_i, X_j)\}_{i,j\in\{1,2,\ldots,n\}}$, across all pairs of features in the training set. We will return to this shortly.

Soft margin and the control of overfitting

As already noted, for any consistent data set and sufficiently large d', there exists an embedding $\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$ such that the embedded data, $\{(\phi(X_i), Y_i)\}_{i=1:n}$, is linearly separable. But this can lead to convoluted (see Figure 1) or unreliable (see Figure 5) decision boundaries.

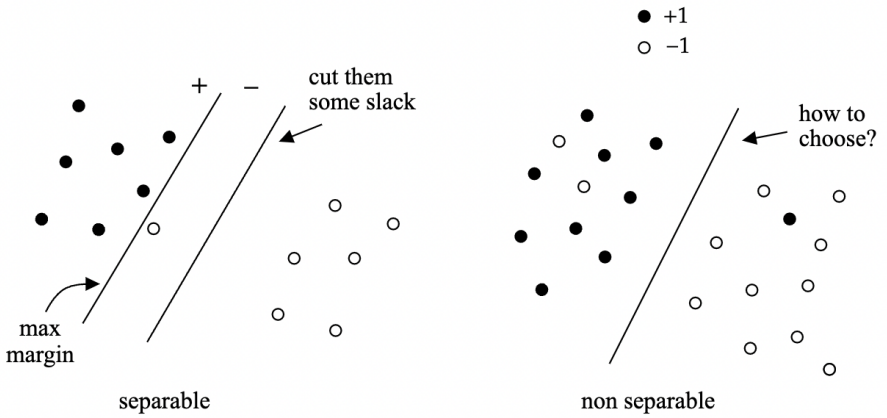


Figure 5: **Soft-margin classifier.** Data depicted in the left-hand panel is linearly separable, but nevertheless the maximum-margin classifier may not be robust; the location of the decision boundary is quite sensitive to what might well be a single outlier. Alternatively, what if the data is not linearly separable? As depicted in the right-hand panel, there may still be a convincing place to locate a linear decision surface. Both of these examples argue for relaxing the maximum-margin criteria to allow for "slack."

It is perhaps a mistake to demand zero errors on the training set, instead of allowing some of the vectors to be misclassified. We can formulate a "soft-margin" classifier by introducing "slack variables," $\xi_1, \xi_2, \cdots, \xi_n$, and then replacing $(**)$ by $(***)$. See below.

$$(\widehat{\alpha}, \widehat{\beta}, \widehat{\xi}) = \text{argmin } (\frac{1}{2}\sum_{i=1}^{d} \beta_i^2 + C\sum_{k=1}^{n} \xi_k) \qquad (***)$$

$$\text{subject to } \xi_k \geq 0 \text{ and } Y_k(\alpha + \beta \cdot X_k) \geq 1 - \xi_k \quad \forall k = 1 : n$$

C determines the penalty or cost of slack:

large C: little slack, less tolerance of errors on training set, more variability

small C: more slack, more tolerance of errors, less variability

As $C \to \infty$, the soft-margin classifier becomes more and more like the maximum-margin classifier (to which it converges in the case of linearly separable data).

Solving $(***)$ is much like solving $(**)$. We play the same game:

(i) introduce lagrange multiplies: $\lambda = \lambda_{1:n}, \gamma = \gamma_{1:n}$ (there are now $2n$ constraints)

(ii) solve for α, β, ξ as functions of λ and γ, $\alpha = \alpha(\lambda, \gamma)$, $\beta = \beta(\lambda, \gamma)$, $\xi = \beta(\lambda, \gamma)$

(iii) solve a max-min problem analogous to $(***)$

Here is the final (maximization) step:

$$\widehat{\lambda} = \text{argmax}\,[\sum_{i=1}^{n} \lambda_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} Y_i Y_j \lambda_i \lambda_j X_i \cdot X_j] \qquad (*****)$$
$$\text{subject to } \lambda_i \in [0, C] \text{ and } \textstyle\sum_{i=1}^{n} Y_i \lambda_i = 0$$

which I include only to note that

- The slack variables ξ_i, $i = 1 : n$ disappeared
- $\widehat{\lambda}$ depends on the feature vectors only through their inner products

$$\{X_i \cdot X_j\}_{1 \le i,j \le n} = \{\kappa(X_i, X_j)\}_{1 \le i,j \le n}$$

The result is identical to the solution to $(***)$ for the maximum-margin classifier, except that $\widehat{\lambda}_i \ge 0$ is replaced by $\widehat{\lambda}_i \in [0, C]$ and $\widehat{\lambda}_i \in (0, C) \Rightarrow \widehat{\alpha}_i = Y_i - \widehat{\beta}_i \cdot X_i$.

The value of λ will be different, but the formula for h is identical:

$$h(x) = \text{sign}(\widehat{\alpha} + \sum_{i=1}^{n} \widehat{\lambda}_i Y_i k(X_i, x))$$

d. Embedding and the kernel trick

We have seen that the weights, $\{\lambda_i\}_{i=1:n}$, in the SVM classifier (with or without slack variables), given a training set (X_i, Y_i), $i = 1 : n$, depend only on

$$\kappa(X_i, X_j) \doteq X_i \cdot X_j \quad \forall i, j = 1 : n$$
$$\text{and } \kappa(X_i, x) \doteq X_i \cdot x \quad \forall x \in \mathbb{R}^d, i = 1 : n$$

What is more, we could just as easily build the SVM classifier using any embedding

$$\phi : \mathbb{R}^d \to \mathbb{R}^d$$
$$\phi(x) = (\phi_1(x), \cdots, \phi_d(x))$$

by simply replacing

$$(X_i, Y_i)_{i=1:n} \text{ with } (\phi(X_i), Y_i)_{i=1:n}$$
$$\kappa(X_i, X_j) = X_i \cdot X_j \text{ with } \kappa(X_i, X_j) = \phi(X_i)\phi(X_j)$$
$$\kappa(X_i, x) = X_i \cdot x \text{ with } \kappa(X_i, x) = \phi(X_i)\phi(x)$$

In short, just replace the kernel function $\kappa(x, \tilde{x}) = x \cdot y \ \forall x, y \in \mathbb{R}^d$ by $\kappa(x, \tilde{x}) = \phi(x) \cdot \phi(\tilde{x}) \ \forall x, y \in \mathbb{R}^d$

So why bother with an embedding? Why not just *start* with some kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$?

The problem is that the computations that gave us maximum margins and optimal soft margins make use of the embedding function ϕ, but not every κ can be written as an inner product. Given a function $\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, we would need to assume that it could be represented in the form $\kappa(x, \tilde{x}) = \phi(x) \cdot \phi(\tilde{x})$ for some $\phi : \mathbb{R}^d \to \mathbb{R}^{'}$ for some d'.

Which raises the question: for which mappings $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ does there exist $\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$ such that

$$\kappa(x, \tilde{x}) = \phi(x)\phi(\tilde{x})$$

There are two key properties:

1. (symmetry)
   Definition: $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is symmetric if $\kappa(x, \tilde{x}) = \kappa(\tilde{x}, x)$ for all $x, \tilde{x} \in \mathbb{R}^d$.

2. (definiteness)
   Definition: Given any m and any distinct $x_1, x_2, \cdots, x_m \in \mathbb{R}^d$, the $m \times m$ matrix
   $$K = \{\kappa(x_i, x_j)\}_{i,j \in \{1,2,\cdots,m\}}$$
   is called the Gram Matrix of κ.

   Definition: If for every m and every distinct $x_1, \cdots, x_m$ the Gram matrix of κ satisfies
   $$a^T K a > 0 \ \ \forall a \in \mathbb{R}^m \ni a \neq 0$$
   then κ is positive definite. If
   $$a^T K a \geq 0 \ \ \forall a \in \mathbb{R}^m \ni a \neq 0$$
   then κ is non-negative definite (aka positive semi-definite).

Symmetric and non-negative definite are pretty much all you need. Loosely:

Meta Theorem If κ is symmetric and non-negative definite, then there exists a function $\phi : \mathbb{R}^d \to \mathbb{R}^{d'} \ni$

$$\kappa(x, \tilde{x}) = \phi(x) \cdot \phi(\tilde{x})$$

"meta" here means that rigorous results like this exist in a large number of contexts and levels of generality:

— diagonalization of symmetric non-neg def matrices

— Bochner's theorem for $\kappa(x, \tilde{x}) = \kappa(x - \tilde{x})$, $x, \tilde{x} \in \mathbb{R}$

— Spectral decomposition theorems in Hilbert and Banach spaces

— Mercer's theorem (see below)

And the applications are ubiquitous:

— Prob & Stat (PCA, Covariance Matrix, Spectral Theory of Stochastic Processes)

— PDEs & Sturm-Liouville Theory

— Fourier Series & Transforms

— Quantum Mechanics (theory of observables)

— Graph Laplacians and graphical approaches to data analysis

Examples

1. (the inner-product kernel) $\kappa(x, \tilde{x}) = x \cdot \tilde{x}$, $\forall x, \tilde{x} \in \mathbb{R}^d$. Is this symmetric? Yes, obviously.

   As for non-neg definite, let $x(1), \cdots, x(m)$ be m distinct vectors in $\mathbb{R}^d$ (think of them as column vectors), and let K be the corresponding Gram matrix. For any $a \in \mathbb{R}^m$, $a \neq 0$:

$$\begin{aligned} a^T K a &= a^T \{x(i) \cdot x(j)\}_{i,j=1:m} a \\ &= \sum_i \sum_j a_i x(i)^T x(j) a_j \\ &= \sum_i a_i x(i)^T \sum_j a_j x(j) \\ &= (\sum_i a_i x(i))^T \sum_j a_j x(j) \\ &= \| \sum_i a_i x(i) \|^2 \geq 0 \end{aligned}$$

   Hence k is non-negative definite.

   What about pos def? Reason this way: Choose $m > d$. Then $x(1), \cdots, x(m)$ *cannot* be linearly independent. Hence for some $a \in \mathbb{R}^m$, $a \neq 0$, $\sum_{i=1}^m a_i x(i) = 0$. But then $Ka = 0$, and hence so does $a^T K a$. K is not positive definite.

2. (finite-dimensional embeddings) Repeat the previous example, but with an embedding:

$$\kappa(x, \tilde{x}) = \phi(x) \cdot \phi(\tilde{x}) \;\; \phi : \mathbb{R}^d \to \mathbb{R}^{d'}$$

   Then by the same argument, κ is non-negative definite but not positive definite, for *any* $d' < \infty$. Hence, no kernel represented by a finite dimensional embedding can be positive definite!

3. (non uniqueness) $\kappa(x, \tilde{x}) = \phi(x) \cdot \phi(\tilde{x})$ does not uniquely define ϕ : Given $x = (x_1, x_2) \in \mathbb{R}^2$, consider

$$\phi(x) = (\phi_1(x), \phi_2(x)) = (x_1, x_2) \quad (\text{So } \phi(x) = x, \text{ the identity map})$$

and let

$$\psi(x) = (\psi_1(x), \psi_2(x)) = \left(\frac{(x_1 + x_2)}{\sqrt{2}}, \frac{(x_1 - x_2)}{\sqrt{2}} \right)$$

and define $\kappa^{(1)}(x, \tilde{x}) = \psi(x) \cdot \psi(\tilde{x})$ and $\kappa^{(2)}(x, \tilde{x}) = \phi(x) \cdot \phi(\tilde{x})$.
Then

$$\kappa^{(1)}(x, \tilde{x}) = x \cdot \tilde{x} = x_1 \tilde{x}_1 + x_2 \tilde{x}_2$$

and

$$\begin{aligned} \kappa^{(2)}(x, \tilde{x}) &= \left(\frac{(x_1 + x_2)}{\sqrt{2}}, \frac{(x_1 - x_2)}{\sqrt{2}} \right) \cdot \left(\frac{(\tilde{x}_1 + \tilde{x}_2)}{\sqrt{2}}, \frac{(\tilde{x}_1 - \tilde{x}_2)}{\sqrt{2}} \right) \\ &= x_1 \tilde{x}_1 + x_2 \tilde{x}_2 \end{aligned}$$

4. (factoring κ) Consider $\kappa(x, \tilde{x}) = (1 + x \cdot \tilde{x})^2$, $x, \tilde{x} \in \mathbb{R}^2$. Is there an embedding function that represents κ? If so, can you find one? Obviously, κ is symmetric. As for non-negative definite, it would suffice to find an embedding function. Expand and look for a factoring:

$$\begin{aligned} \kappa(x, \tilde{x}) &= (1 + x_1 \tilde{x}_1 + x_2 \tilde{x}_2)^2 \\ &= 1 + 2x_1 \tilde{x}_1 + 2x_2 \tilde{x}_2 + x_1^2 \tilde{x}_1^2 + x_2^2 \tilde{x}_2^2 + 2x_1 \tilde{x}_2 x_2 \tilde{x}_2 \\ &= \phi(x) \cdot \phi(\tilde{x}) \end{aligned}$$

where

$$\phi(x) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1 x_2) \in \mathbb{R}^6$$

But, κ cannot be positive definite, since $d' = 6 < \infty$.

5. (Radial basis function) Fix a dimension $d \geq 1$, and consider $\kappa(x, \tilde{x}) = e^{-\gamma \|x - \tilde{x}\|^2}$, for any $x, \tilde{x} \in \mathbb{R}^d$, for some $\gamma > 0$. This is knowns and the "radial basis function", and it turns to be positive definite. The proof is not trivial (see HW, 2610).

6. (Mercer's Theorem) Finally, here is a specific example of the "meta theorem":

   Mercer's Theorem If $\kappa : [a, b]^d \times [a, b]^d \to \mathbb{R}$ is continuous, symmetric and non-negative definite, then there exists d' (finite or infinite) and $\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$ such that

$$\kappa(x, \tilde{x}) = \phi(x) \cdot \phi(\tilde{x})$$

   Remarks:

   (a) Mercer's Theorem is more typically stated as a "spectral representation":

   under the same assumptions

$$\kappa(x, \tilde{x}) = \sum_{i=1}^{\infty} \lambda_i e_i(x) e_i(\tilde{x}) \tag{1}$$

   where $\lambda_i \geq 0 \ \forall i$ (wlog $\lambda_1 \geq \lambda_2 \geq \cdots$) and $e_1, e_2, \cdots$ are orthonormal and constitute a basis for the set of functions on $[a, b]^d$.

In fact, e_i is an "eigenvector" of k with eigenvalues λ_i, in the sense that

$$\int_{\tilde{x}\in[a,b]^d} \kappa(x,\tilde{x})e_i(\tilde{x})d\tilde{x} = \lambda_i e_i(x)$$

and "orthonormal" means

$$\int_{y\in[a,b]^d} e_i(y)e_j(y)dy = \delta_{ij}$$

(b) In this representation, any (sufficiently regular) function f on $[a,b]^d$ can be written as

$$f(x) = \sum_{i=1}^{\infty} a_i e_i(x)$$

(The coefficients, $a_1, a_2, \cdots$, are sometimes called the "generalized Fourier transform" of f. More simply, they represent f in a rotated coordinate system).

(c) k is positive definite iff $\lambda_i > 0$ for all i

(d) From the spectral representation (1), we can "read off" an embedding:

$$\phi(x) \doteq (\sqrt{\lambda_1}e_1(x), \sqrt{\lambda_2}e_2(x), \sqrt{\lambda_3}e_3(x), \ldots)$$

Or, in other words, $\phi_l(x) = \sqrt{\lambda_l}e_l(x),\ \forall l = 1, 2, \ldots$

e. Positive definite kernels

We have already seen that $d' = \infty$ is a necessary condition for

$$\kappa(x,\tilde{x}) = \phi(x)\phi(\tilde{x}), \phi : \mathbb{R}^d \to \mathbb{R}^{d'}$$

to be positive definite—see examples (1) and (2), above. But why do we care about the distinction between positive definite and non-negative definite? Consider $d' = 2$ and notice that any set of 4 distinct points has a labeling that is *not linearly separable*! Hence there is *no embedding* into $\mathbb{R}^2$ that will admit perfect linear separation for all data sets with four or more points ($n \geq 4$).

More generally, in $\mathbb{R}^d$, any set of $n \geq d+2$ points can be labeled in a way that makes it non linearly separable. (This relates to "complexity theory", "the Vapnik-Chervonenkis dimension", and the beginning of a systematic approaches to non-parametrics that emerged from Russia in the 1960s.)

But the situation for positive definite kernels is entirely different:

Proposition. Let k be a positive definite kernel. Then for any data $\{(X_i, Y_i)\}_{i=1:n}$, with distinct features $X_1, \ldots, X_n$, there exists $\lambda \in \mathbb{R}^n$ such that

$$h(x) = \text{sign}(\sum_{i=1}^{n} \lambda_i \kappa(X_i, x))$$

correctly classifies the data, i.e.

$$Y_j = \text{sign}(\sum_{i=1}^{n} \lambda_i \kappa(X_i, X_j))$$

Pf:

It would be sufficient to solve the n equations:

$$\sum_{i=1}^{n} \lambda_i \kappa(X_i, X_j) = Y_j \quad j = 1 : n$$

for λ, since then we wouldn't even need the sign function: $h(x) \doteq \sum_{i=1}^{n} \lambda_i \kappa(X_i, x)$ would correctly classify every example in the data.

Since κ is symmetric, it is sufficient, alternatively, to solve

$$\sum_{1}^{n} \kappa(X_j, X_i) \lambda_i = Y_j$$

for λ. Succinctly, we want to show that

$$K\lambda = Y$$

always has a solution, where K is a Gram matrix. But this is a linear system, and it has a solution iff K is invertible, i.e. iff

$$Ka \neq 0 \ \ \forall a \in \mathbb{R}^n, a \neq 0$$

which follows from the positive definite property:

$$a^T K a > 0 \ \ \forall a \in \mathbb{R}^n, a \neq 0$$

II Some statistical theory

C Building classifiers

Deep neural network classifiers

a. Architecture and notation
b. Loss function and training
c. Four surprises
d. Softmax and maximum-margin classification
e. Zero errors

a. Architecture and notation. See figure (1), which depicts a basic architecture of a typical neural network designed to solve the classification problem. As usual, we will use $x \in \mathbb{R}^d$ to represent a feature vector, which is typically some pre-processed version of an observable. The prototypical example is the image classification problem, in which case the input could either be raw pixel intensities or intensities following some kind of contrast enhancement or standardizations, and possibly also including some data-reduction step amounting to a kind of sub-sampling. We will focus on the problem of labeling a scene as belonging to one of s categories, which could be as simple as a binary indicator for the presence or absence of a particular kind of object, or a much more elaborate characterization involving thousands of potential objects, object parts, and object groupings.

As suggested by the figure, we can think of the network as representing two processing steps, the first of which amounts to a feature embedding and the second of which amounts to classification by softmax, using the embedded data. The overwhelming bulk of machinery is devoted to the embedding, $\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$. This multi-layered portion of the architecture is parameterized by a typically enormous parameter vector, $w \in \mathbb{R}^m$, with millions, or billions, or occasionally trillions of components. (We will often write $\phi(x) = \phi(x; w)$ to emphasize the parametrization of the embedding.) These parameters represent connection strengths among idealized neurons, which are themselves organized in tens or hundreds of discrete layers. A layer may contain hundreds or thousands of neurons, each of which has inputs from other neurons in the same layer or one or two layers below, and outputs to other neurons in the same layer or one or two layers above.

The softmax is nothing more than the discriminative classifier that we studied earlier (§II.B), but applied to the embedded data, i.e. the outputs of the units from the penultimate layer. We use the vector $\phi(x) = (\phi_1(x), \ldots, \phi_{d'}(x))$ to represent these outputs, which are, in turn, the inputs to the last, "linear" layer. Here, the inner products of the embedded feature vector with each of s weight vectors, $\beta_1, \ldots, \beta_s \in \mathbb{R}^{d'}$, are computed. These, then, are the s arguments of the softmax function, and the result is an estimate of the conditional probability distribution over the s categories. In sum, the transformation from input to output is parameterized by the $m + sd'$ parameters $w_1, \ldots, w_m, \beta_1, \ldots, \beta_s$.

In virtually every aspect, the network amounts to an absurd model of biological brains. Nonetheless, given enough training data the performance can be spectacular.

b. Loss functions and training. We build a support-vector machine by first *choosing* an embedding (via the kernel trick), and then learning (aka "fiting," aka "estimating") a margin classifier on the embedded

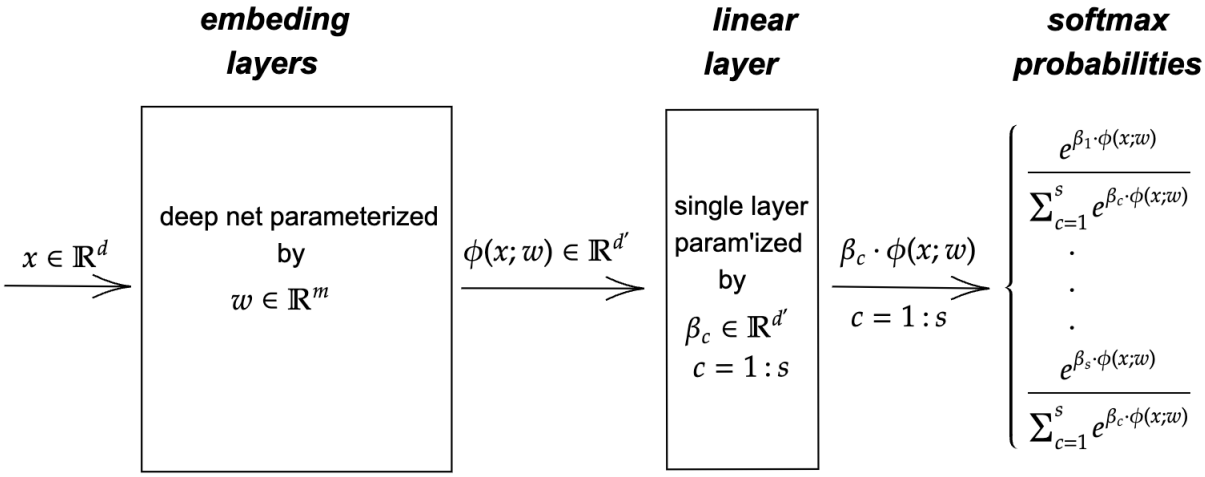


Figure 1: **Neural Net from the Embedding Viewpoint.** The deep net serves as an excellent embedding of the feature space, $x \to \phi(x; w)$. In fact, it is not uncommon to find, after training, that the *embedded data* $(\phi(x_k; w), y_k),\ k = 1 : n$ is linearly separable.

features.The neural network, on the other hand, uses the softmax classifier and learns *both* an embedding and the classifier simultaneously.

Specifically, learning is by a variant of gradient descent ("stochastic gradient descent"—discussed later) of the likelihood function. Put aside embedding, for the moment, and recall the softmax model:

$$r_c(x) = \mathbb{P}(Y = c|x) = \mathbb{E}[\mathbb{1}_{Y=1}\,|x] = \frac{e^{\beta_c \cdot x}}{\sum_{\tilde{c}=1}^{n} e^{\alpha_{\tilde{c}} + \beta_{\tilde{c}} \cdot x}} \quad c = 1 : s$$

(Here we have used the over-parametrized representation—discussed in an earlier homework. Also, since we can always assume that one of the features is a fixed constant, we have dropped the explicit "bias" terms $\alpha_c,\ c = 1 : n$.) Given training data (x_k, y_k), $k = 1 : n$, $x_k \in \mathbb{R}^d$ and $y_k \in \{1, \ldots, s\}$, the likelihood function becomes

$$L\left(\{(x_k, y_k)\}_{k=1:n}; \beta_1, \ldots, \beta_s\right) = \prod_{k=1}^{n} \mathbb{P}(Y = y_k|x_k) = \prod_{k=1}^{n} r_{y_k}(x_k; \beta_1, \ldots, \beta_s)$$

Hence the maximum-likelihood estimator is

$$\begin{aligned}
\underset{\beta_1,\ldots,\beta_s}{\operatorname{argmax}} L\left(\{(x_k, y_k)\}_{k=1:n}; \beta_1, \ldots, \beta_s\right) &= \underset{\beta_1,\ldots,\beta_s}{\operatorname{argmax}} \frac{1}{n} \sum_{k=1}^{n} \log r_{y_k}(x_k; \beta_1, \ldots, \beta_s) \\
&= \underset{\beta_1,\ldots,\beta_s}{\operatorname{argmin}} \frac{-1}{n} \sum_{k=1}^{n} \log r_{y_k}(x_k; \beta_1, \ldots, \beta_s) \\
&= \underset{\beta_1,\ldots,\beta_s}{\operatorname{argmin}} - \sum_{c=1}^{s} \hat{p}_c \left(\frac{1}{n_c} \sum_{k:y_k=c} \log r_c(x_k; \beta_1, \ldots, \beta_s) \right) \qquad (1)
\end{aligned}$$

where n_c is the number of training samples labeled c, and $\hat{p}_c$ is the empirical estimator of π_c, the true (*a priori*) probability. The final term, the one being minimized in (1), is known in the machine learning literature as the "cross-entropy loss function." In other words, maximizing likelihood is equivalent to minimizing cross-entropy.

But what about the embedding parameters? How do we use (1) to learn both $\beta_1, \ldots, \beta_s$ *and* w? Just replace (x_k, y_k) by $(\phi(x_k; w), y_k)$, for each $k = 1 : n$, and d by d', in which case $\beta_i \in \mathbb{R}^d$ becomes $\beta_i \in \mathbb{R}^{d'}$, for each $i = 1 : s$. Then, by the same maximum-likelihood principle, *we would seek to minimize*

$$-\sum_{c=1}^{s} \hat{p}_c \left(\frac{1}{n_c} \sum_{k:y_k=c} \log r_c \left(\phi(x_k; w); \beta_1, \ldots, \beta_s\right) \right)$$

over all possible assignments of the $m + sd'$ *parameters* $w_1, \ldots, w_m, \beta_1, \ldots, \beta_s$. We can make this a little more transparent, and the representation more convenient, if we replace $w, \beta_1, \ldots, \beta_s$ by a single parameter vector $W \in \mathbb{R}^{m+sd'}$:

$$W \doteq (w_1, \ldots, w_m, \beta_{11}, \ldots, \beta_{1d'}, \ldots, \ldots, \beta_{s1}, \ldots, \beta_{sd'})$$

The awkward expression $r_c \left(\phi(x_k; w); \beta_1, \ldots, \beta_s\right)$ for $\mathbb{P}(Y = c|x)$ becomes $r_c = r_c(x; W)$, and the cross-entropy loss becomes

$$-\sum_{c=1}^{s} \hat{p}_c \left(\frac{1}{n_c} \sum_{k:y_k=c} \log r_c(x_k; W) \right) \tag{2}$$

or more succinctly

$$-\frac{1}{n} \sum_{k=1}^{n} \log r_{y_k}(x_k; W) \tag{3}$$

As already noted, the computational approach is through a stochastic version of gradient descent. We will have more to say about that later, but for now let's recall the discrete version of (ordinary) gradient descent[1]: after initializing W with W_o, iterate

$$W_t = W_{t-1} - \varepsilon \nabla_W \left(-\frac{1}{n} \sum_{k=1}^{n} \log r_{y_k}(x_k; W) \right) \tag{4}$$

Throughout the procedure we can monitor progress, not only by evaluating the loss function at W_t, but also by recording the *in-sample error rate*:

$$\frac{\#\{k : y_k \neq \operatorname{argmax}_c r_c(x_k; W_t)\}}{n} \tag{5}$$

Remarks.

i. ε is often called the "learning rate." It is typically chosen to decrease to zero as the number of iterations, t, increases to infinity.

[1]This is the process we used previously for computing parameters of the partition function, as well as for finding the maximum-likelihood estimators for exponential families.

ii. "Stochastic Gradient Descent" refers to the introduction of noise into equation (4). There are many ways to do this, but the idea is to inject enough randomness to allow the trajectory, W_t, to escape from local minima, while not so much noise as to overwhelm the slow decrease in the learning rate. (See **e**, "Zero errors.")

The gradient is complicated, involving top-down, level-by-level, recursive calculations starting with the loss function and ending with first-layer terms that directly depend on the input x, as is required by virtue of the chain rule. An amazing array of hardware and software improvements have come together to make this feasible.

c. Four surprises.

1. **First surprise: perfect fit.** Usually, settings for the various parameters that specify the stochastic version of gradient descent (sometimes called "hyper parameters," such as the learning rate and the degree of randomness) can be found so that the in-sample error rate (5) converges to zero; training leads to perfect classification of the sample data. Ordinarily, we would expect a penalty for building a model that was this faithful to the data, but instead we often find very good generalization performance.

   It is tempting to interpret this observation as reflecting, at least in part, regularity properties of real data. But it turns out that this property alone is not very distinguishing, *since zero errors is still achievable when training on data that is first corrupted by a random permutation of the sample labels*: $y_i \to y_{\sigma(i)}$, where σ is a random permutation, $\sigma : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$, chosen from the uniform distribution on all $n!$ possible permutations!

2. **Second surprise: consequences of the implicit function theorem.** This theorem makes precise a common intuition: if I have more unknown variables than equations constraining these variables, then there will be more than one solution. In fact, we can make some fairly precise statements about the set of all solutions, but first let's look at some examples:

   (i) **(two degrees of freedom; one constraint)** Suppose that there are two parameters (call them x and y) but just one equation: $x^2 + y^2 = 1$. Notice that if (x_o, y_o) is a solution, with $x_o \notin \{-1, 1\}$, then you could move continuously from (x_o, y_o) to nearby solutions by making small changes in x and then solving for the change in y. If the changes in x were to bring x to ± 1, it would still be just as easy to move continuously on the solution set by making small changes in y and solving for x. In other words, for any starting point we can describe one of the variables, at least *locally*, as a smooth function of the other:

$$y = \begin{cases} \sqrt{1 - x^2} & \text{for } x \in (-1, 1), \text{ or} \\ -\sqrt{1 - x^2} & \text{for } x \in (-1, 1) \end{cases}$$

   and/or

$$x = \begin{cases} \sqrt{1 - y^2} & \text{for } y \in (-1, 1), \text{ or} \\ -\sqrt{1 - y^2} & \text{for } y \in (-1, 1) \end{cases}$$

   Think of this in terms of degrees of freedom: we start with two, one for each parameter, and then subtract one for each constraint. We end up with a curve, which is an object with one degree of freedom. We could summarize the situation, geometrically, by saying that the set of solutions define a smooth one-dimensional manifold.

(ii) **(three degrees of freedom; two constraints)** Suppose that there are three parameters—call them x, y, and z, and two equations:

$$x^2 + y^2 + z^2 = 1$$
$$x + y + z = 0$$

We start with three degrees of freedom, subtract one for each constraint, and end up with one degree of freedom. So the set of solutions should again define a smooth one-dimensional manifold. Analytically, we would expect to be able to solve for any pair of the three variables, say y and z, in terms of the third variable, say x:

$$y^2 + z^2 = \pm\sqrt{1 - x^2}$$
$$y + z = -x$$

To really understand the solution set we would need to solve the two equations for the pair, $(y(x), z(x))$, for each $x \in (0, 1)$. This is a little messy and perhaps not the best way to visualize the solutions set, which we already knew was a smooth one-dimensional manifold. Alternatively, perhaps you can describe the manifold without solving anything? Check out the footnote for a spoiler[2].

Let's look at one more example, before examining its implications for our classifier neural-network:

(iii) **(three degrees of freedom; one constraint)** Same situation, but without the additional constraint:

$$x^2 + y^2 + z^2 = 1$$

We start with three degrees of freedom and subtract one for the single constraint, ending up with two. If we solve for (say) z in terms of x and y, then the solution set should describe a smooth two-dimensional manifold. Indeed,

$$z = \begin{cases} \sqrt{1 - x^2 - y^2} & \text{for } (x, y) \in (-1, 1) \times (-1, 1), \text{ or} \\ -\sqrt{1 - x^2 - y^2} & \text{for } (x, y) \in (-1, 1) \times (-1, 1) \end{cases}$$

Neither of the two pieces (each a hemisphere) describes the entire manifold, and in fact the two together still leave the surface unspecified at the domain boundaries, where $x^2 + y^2 = 1$ and $z = 0$. In general, and as in the first example, a complete specification requires piecing together different representations.

The surprise comes when we apply this reasoning to an over-parameterized neural network: Suppose we stop training after t iterations, at which point we have the particular estimate $\hat{W} = W_t$. For each $k = 1 : n$, let

$$l_k = -\log r_{y_k}(x_k; \hat{W})$$

Notice that we can then write the cross-entropy loss as $\frac{1}{n}\sum_{k=1}^{n} l_k$. Now define $D = w + sd'$, the number

[2]The solution is the intersection of a sphere and a plane through the origin, and hence it is a "great circle" of the sphere, i.e. a circle on the sphere with the same radius as the sphere. Like any of the longitudes, but not the latitudes, of the earth.

of parameters in W, and consider the set of all W that satisfy the n equations

$$\begin{aligned}
\log r_{y_1}(x_1; W) + l_1 &= 0 \\
\log r_{y_2}(x_2; W) + l_2 &= 0 \\
&\vdots \\
\log r_{y_n}(x_n; W) + l_n &= 0
\end{aligned}$$

There are n equations and hence n constraints on the D parameters in W, and we should therefore expect that the set of all W satisfying these D constraints will define an $D - n$ dimensional manifold. Observe that the cross-entropy loss, which is just the sum $\sum_{k=1}^{n} l_k$, will be invariant on this manifold. To illustrate, with some numbers, suppose that there are one-hundred billion parameters ($D = 10^{11}$), and ten-million examples in the training set ($n = 10^7$). Then the are $10^{11} - 10^7 = 99,990,000,000$ directions in which we could depart from $\hat{W}$ without changing the value of the loss function on the training set. It would appear to be very unlikely that all of these alternative sets of parameters will be equally effective in their out-of-sample performance (i.e. the ability to generalize beyond the training data). How is it that training not only finds manifolds with low in-sample loss, but also selects parameters within these manifolds that lead to good generalization?

If you are interested in the precise treatment of these degrees-of-freedom type arguments, then you should start with the implicit function theorem:

> **Implicit Function Theorem.** Given two positive integers n_1 and n_2, let $\boldsymbol{x} = (x_1, \ldots, x_{n_1}) \in \mathbb{R}^{n_1}$, $\boldsymbol{y} = (y_1, \ldots, y_{n_2}) \in \mathbb{R}^{n_2}$, and write $(\boldsymbol{x}, \boldsymbol{y})$ to mean $(x_1, \ldots, x_{n_1}, y_1, \ldots, y_{n_2}) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} = \mathbb{R}^{n_1+n_2}$. Let $F = F(\boldsymbol{x}, \boldsymbol{y}) : \mathbb{R}^{n_1+n_2} \to \mathbb{R}^{n_2}$ be continuously differentiable.
>
> Assume that for some $\boldsymbol{x} = \boldsymbol{x}_o$ and $\boldsymbol{y} = \boldsymbol{y}_o$, $F_k(\boldsymbol{x}_o, \boldsymbol{y}_o) = 0, \forall k = 1 : n_2$, and let J be the $n_2 \times n_2$ Jacobian matrix $\frac{\partial F}{\partial \boldsymbol{y}}$ evaluated at $(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}_o, \boldsymbol{y}_o)$:
>
> $$J = \{J_{ij}\}_{i,j \in 1:n_2} = \left\{ \frac{\partial F_i}{\partial y_j} \right\}_{i,j \in 1:n_2}$$
>
> Finally, assume that J is invertible at $(\boldsymbol{x}_o, \boldsymbol{y}_o)$. Then there exists a continuously-differentiable function $f : \mathbb{R}^{(n_1)} \to \mathbb{R}^{(n_2)}$, defined on an open set U containing $\boldsymbol{x}_o$, such that $F(\boldsymbol{x}, f(\boldsymbol{x})) = 0$ for all $\boldsymbol{x} \in U$.

To connect to the example, take $n_1 = D - n$, $n_2 = n$, and $F_k(W) = r_{y_k}(x_k; W) + l_k,\ k = 1 : n$.

3. **Third surprise: the double dip.** One would expect that estimating more parameters than there are examples would be a bad idea. But the story turns out to be more complicated than the traditional tradeoff between bias and variance—see figure (2), which reveals several common aspects of the relationship between out-of-sample (aka generalization) performance and the dimensionality D of the parameter space.

   The dark (continuous) curve represents the error rate on the test (as opposed to training) data, plotted as a function of the number of parameters in the network. Each point on the dark line represents the performance of a nearly fully trained network, meaning that there have been enough iterations of the gradient update (4) that the negative of the log-likelihood is very nearly flat.

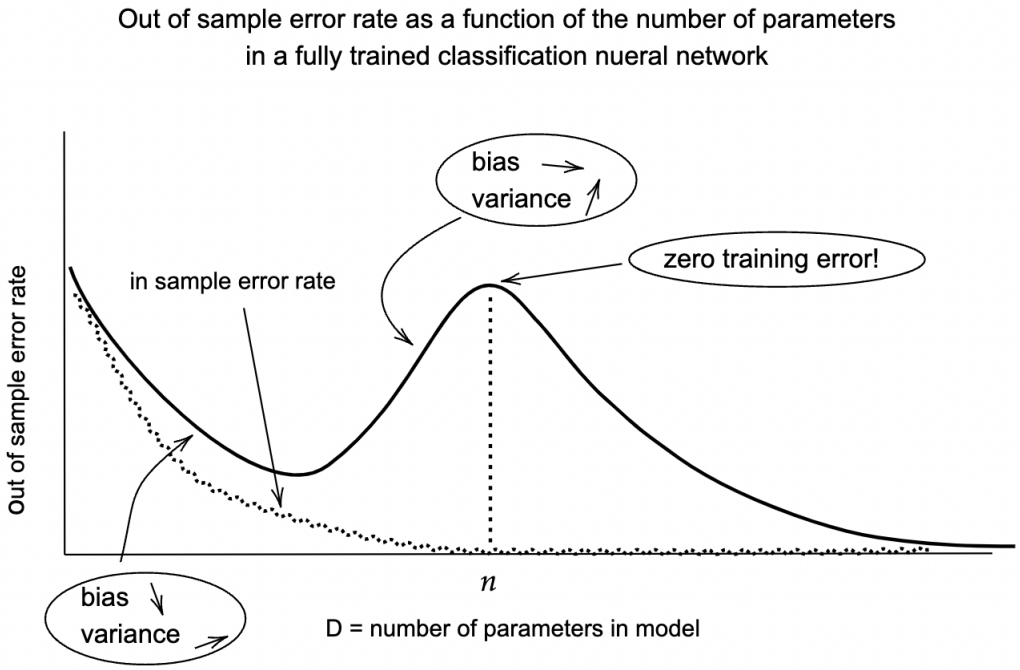


Figure 2: **Double dip.** Solid curve is the out-of-sample error rate of a fully-trained DNN with D parameters and a training sample of size n. The irregular, dotted, curve is the corresponding in-sample error rate, which is essentially zero when there are more parameters, D, then there are samples, n, in the training set. See text.

As the number of parameters is increased, the out-of-sample performance improves (number of errors decrease). This is pretty much a classical case of rapidly decreasing bias (the model is getting more flexible and can accommodate richer structures in the data) and slowly increasing variance (more parameters). There is a sweat spot, where the bias-variance tradeoff is balanced, which is evidently at the bottom of the first dip. Further increases in model complexity (number of parameters) incur rapidly increasing variance with slower increases in bias. Performance deteriorates.

The surprise is the second dip, which follows a peak in out-of-sample error occurring almost exactly when the number of parameters D is equal to the number of samples n. The data is perfectly fit, in the sense that the in-sample error rate is approximately (and often exactly) zero. The data has been overfit, and one would expect that further increasing D would lead to increasing variance with little or no decrease in bias, exacerbating the poor out-of-sample performance. But the opposite occurs. The out-of-sample performance continues to improve. Why?

For that matter, given that the in-sample error rate is already zero when $D = n$, why is there *any* difference among the models with $D \geq n$? How do additional parameters play into the resulting fully trained models? The answers are not fully known, despite intense research over the past decade. But some clues have emerged that may have taken away some of the mystery. Two of these are the phenomena called "neural collapse" (the fourth surprise—see below) and an unanticipated connection between the softmax and the maximum-margin classifiers (see paragraph (d), also below).

We need to keep in mind that training is based on improving *likelihood* (minimizing *cross-entropy*), which is a very different criteria from minimizing *sample error rate*. But they are not unconnected. Perhaps the best way to think about it is to imagine that you are hanging out in the penultimate layer of the network,

i.e. the d'-dimensional space of the embedded feature data, $\phi(x;w) = (\phi_1(x;w), \ldots, \phi_{c'}(x;w)) \in \mathbb{R}^{d'}$). What can we say about the arrangement of the particular n vectors? Here are some observations:

(1) Zero errors (perfect classification of the training data) only occurs if the embedded data is linearly separable. Because:

$$\begin{aligned}
\text{zero error rate} \Leftrightarrow\ & \forall k = 1:n, \quad \operatorname*{argmax}_c r_c\left(\phi(x_k;w);\beta_1,\ldots,\beta_s\right) = y_k \\
\Leftrightarrow\ & \forall k = 1:n, \quad r_{y_k}\left(\phi(x_k;w);\beta_1,\ldots,\beta_s\right) > r_c\left(\phi(x_k;w);\beta_1,\ldots,\beta_s\right) \quad \forall c \neq y_k \\
\Leftrightarrow\ & \forall k = 1:n, \quad \frac{e^{\beta_{y_k}\cdot\phi(x_k;w)}}{\sum_{\tilde{c}=1}^{s} e^{\beta_{\tilde{c}}\cdot\phi(x;w)}} > \frac{e^{\beta_c\cdot\phi(x_k;w)}}{\sum_{\tilde{c}=1}^{s} e^{\beta_{\tilde{c}}\cdot\phi(x;w)}} \quad \forall c \neq y_k \\
\Leftrightarrow\ & \forall k = 1:n, \quad \beta_{y_k}\cdot\phi(x_k;w) > \beta_c\cdot\phi(x_k;w) \quad \forall c \neq y_k
\end{aligned} \tag{6}$$

(2) Once the embedded data is linearly separable, scaling up the coefficients in the linear layer, $\beta_1,\ldots,\beta_s \to \alpha\beta_1,\ldots,\alpha\beta_s$ for any $\alpha > 1$, increases the likelihood:

$$\begin{aligned}
L(\{x_k,y_k\}_{k=1:n};w,\beta_1,\ldots\beta_s) &= \prod_{k=1}^{n} \mathbb{P}(Y = y_k \mid x_k;W) \\
&= \prod_{k=1}^{n} \frac{e^{\beta_{y_k}\cdot\phi(x_k;w)}}{\sum_{c=1}^{s} e^{\beta_c\cdot\phi(x_k;w)}} \\
&= \prod_{k=1}^{n} \frac{1}{\sum_{c=1}^{s} e^{(\beta_c-\beta_{y_k})\cdot\phi(x_k;w)}} \\
&= \prod_{k=1}^{n} \frac{1}{1+\sum_{c\neq y_k}^{s} e^{(\beta_c-\beta_{y_k})\cdot\phi(x_k;w)}} \\
&< \prod_{k=1}^{n} \frac{1}{1+\sum_{c\neq y_k}^{s} e^{\alpha(\beta_c-\beta_{y_k})\cdot\phi(x_k;w)}} \quad \text{(by virtue of (6))} \\
&= L(\{x_k,y_k\}_{k=1:n};w,\alpha\beta_1,\ldots\alpha\beta_s)
\end{aligned}$$

(3) Notice that scaling actually increases the separation between categories, in the sense that for all k, $c \neq y_k$, and $\alpha > 1$

$$\frac{r_{y_k}(\phi(x_k;w);\beta_1,\ldots,\beta_s)}{r_c(\phi(x_k;w);\beta_1,\ldots,\beta_s)} > \frac{r_{y_k}(\phi(x_k;w);\alpha\beta_1,\ldots,\alpha\beta_s)}{r_c(\phi(x_k;w);\alpha\beta_1,\ldots,\alpha\beta_s)}$$

But, be careful: this only works if the data is already linearly separated (i.e. if the performance has already achieved zero in-sample errors); otherwise the contributions to the likelihood of some of the terms could go down.

4. **Fourth surprise: "Neural collapse."**

We have already mentioned that it is frequently observed that training a DNN with $D > n$ leads to zero training errors. It has also been observed in several laboratories (Papyan et al., PNAS V1117, 2020, seems to have been the first), that additional iterations of stochastic gradient descent on the cross-entropy (maximum likelihood) loss often leads to a canonical arrangement of the embedded feature vectors (see figure **??**). In this arrangement, the embedded features belonging to each class, $c \in \{1,2,\ldots,s\}$, converge to their class means. Since there are s categories, these means define an $s-1$ dimensional linear subspace in the embedding space $\mathbb{R}^{d'}$.

The figure depicts the two-dimensional subspace defined by the class means from the $s = 3$ categories. The small colored circles represent the projections of the features into the two-dimensional plane defined by the three class means. Each color represents one category. The top panel illustrates the arrangement at a relatively early stopping point in the gradient procedure; the bottom panel illustrates a more fully trained arrangement. As the training continues, the three means become maximally separated and the within-class spread of the embedded features, relative to the between-class spreads of the means, goes to zero— a process dubbed "neural collapse" by Papyan et al. These and other characteristics emerge from over-parametrized neural networks that continue to be trained after achieving zero training errors, and are consistent with the "second dip" depicted in figure (3).

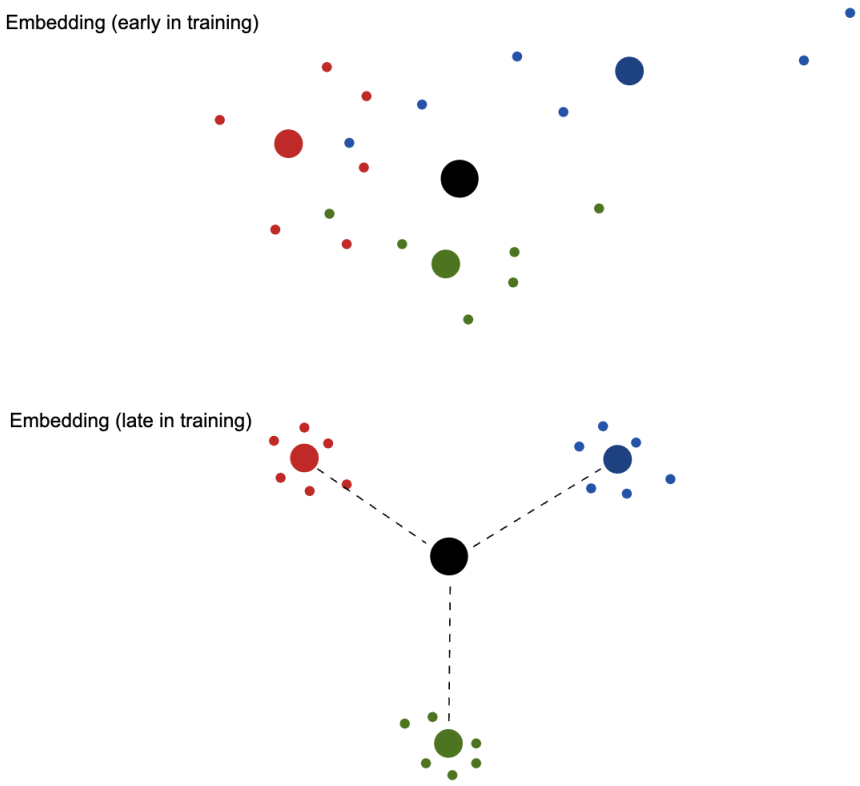


Figure 3: **Neural collapse.** The feature vectors are embedded in a d'-dimensional vector space, but empirically the data collapses to a canonical arrangement in $s - 1$ dimensions. In this depiction, there are $s = 3$ categories and the training data contains 6 examples ($n = 18$) from each category. The different colors represent the different categories. The colored, medium-sized circles contain the sample means of the embedded features from the respective categories. The large black circle is the mean of all n embedded features. In both the top and bottom panels, the small circles represent the projections of the embedded feature vectors onto the two-dimensional linear subspace that contains the three sample means, one for each category. The top panel depicts an arrangement after a relatively small number of (stochastic) gradient steps; the bottom panel depicts the arrangement after a larger number of steps. See text.

d. Softmax and maximum-margin classification.

DNN classifiers are routinely trained with the number of parameters exceeding the number of samples by many orders of magnitude. The implicit function theorem indicates that any performance on the training set can be reproduced, exactly, by any of an infinite collection of parameter vectors—namely, the ones lying on a smooth high-dimensional manifold. Presumably, much of the area of the manifold is taken up by parameter vectors that would neither interpolate nor extrapolate to good out-of-sample performance. By what mechanisms do DNN classifiers avoid choosing poorly performing parameters?

One suggestion, which is backed by some theoretical as well as experimental results, is that the training of certain loss functions via gradient descent can lead to models that extend smoothly beyond the training data, thereby producing a kind of implicit regularization.[3] A wonderful example comes from a new look at a very old approach to classification, the softmax model. One might not have expected that there would be anything fundamentally new to be learned about this time-honored example of the discriminative approach to classification. Nevertheless, Soudry et al. have shown that even deterministic gradient ascent of the log-likelihood function for the softmax model will converge to the maximum-margin classifier when trained on linearly separable data ("The Implicit Bias of Gradient Descent on Separable Data," JMLR 19, 2018, 1-57). The main idea of the proof is that the gradient is exponentially dominated by the contributions from the margin vectors.

The result is relevant since (i) we can always engineer an unambiguous training set to be linearly separable (e.g. with kernel methods as used in support vector machines), and (ii) as we have just discussed, empirical results on DNN classifiers indicate that the highest layers will often feed the final softmax classifier with linearly separable data ("neural collapse").

e. Zero errors. Arguably, the biggest remaining mystery is the frequent convergence to a perfect classification of the training data. As we have seen, at that point the embedded data must be linearly separable. This, combined with the result of Soudry et al., suggests that further steps of the gradient iteration serve to improve the margins separating the categories—an implicit bias that supports good generalization.

Every minimum and every saddle point (let's call them singular points) has a "basin of attraction," a region of starting locations from which gradient descent will converge to that singular point. Absent any reason for picking one starting point versus another, training is usually initialized randomly. Given the complex landscape represented by the cross-entropy loss function (3), we can not expect good performance from a deterministic gradient descent initialized at a random starting point.

If we formulate the problem generically, we find a long history of contributions to approaches that involve *stochastic* gradient descent, mostly from the optimization communities. Randomness might enter because of the nature of the observables (e.g. only a noisy, inexact, or otherwise corrupted version of the gradient

[3]Here are some good starter references on the connections between gradient descent and implicit regularization:

(i) Evidence from Facebook Research that stacking matrices (i.e. layers of fully-connected linear units) in the "bottleneck" of an autoencoder improves performance: (Jing et al.) arXiv:2010.00679v2 [cs.LG] 14 Oct 2020

(ii) A relatively clean special case, and a relatively complete analysis: (Chou et al.) arXiv:2011.13772v3 [cs.LG] 7 Apr 2021

(iii) Empirical and theoretical analysis of implicit regularization in deep *linear* networks: (Aurora et al.) arXiv:1905.13655v3 [cs.LG] 26 Oct 2019

is available) or, by design, for the purpose of exploring other basins. In the case of neural networks, the initial motivation seems to have been an effort to sidestep the computationally intensive problem of visiting every example for every update of the gradient. Why not just take a sampling of the data (a batch) for each step? But it turns out that there are additional benefits deriving from a better exploration of the loss function.

Let's view the problem abstractly. Starting with (3), let us write $g_k(W)$ for $\log r_{y_k}(x_k; W)$, in which case the loss function becomes

$$G(W) = \frac{1}{n}\sum_{k=1}^{n} g_k(W)$$

and (deterministic) gradient descent is just

$$W_t = W_{t-1} - \varepsilon \nabla_W G(W_{t-1})$$

Let $\hat{\nabla}_W G(W)$ be an unbiased estimate of the true gradient, $\nabla_W G(W)$:

$$\mathbb{E}\left[\hat{\nabla}_W G(W)\right] = \nabla_W G(W)$$

Stochastic gradient descent replaces $\nabla_W G$ by $\hat{\nabla}_W$:

$$W_t = W_{t-1} - \varepsilon \hat{\nabla}_W G(W_{t-1}) \tag{7}$$

Stochastic gradient descent: escaping local singularities.

The idea is that a noisy gradient can be better than a noiseless gradient, since in principle the randomness allows the trajectory, W_t, to explore outside of the initial basin of attraction. As an example, suppose that

$$\hat{\nabla}_W G(W) = \nabla_W G(W) - \sigma\eta$$

where $\mathbb{E}[\eta] = 0$. Assume that for every $t = 1, 2, \ldots$, $\eta = \eta_t$, where $\eta_1, \eta_2, \ldots$ are *iid* with common mean zero and variance one. Then the stochastic gradient descent in (7) becomes

$$W_t = W_{t-1} - \varepsilon(\nabla_W G(W_{t-1}) + \sigma\eta_{t-1})$$

There are two parameters, ε and σ. It is convenient to re-parameterize to ε and T (for "temperature")—replace σ by $\sqrt{\frac{2T}{\varepsilon}}$ for $T > 0$:

$$W_t = W_{t-1} - \varepsilon \nabla_W G(W_{t-1}) + \sqrt{2\varepsilon T}\eta_{t-1} \tag{8}$$

It is not hard to find bounds and smoothing conditions on G such that as $t \to \infty$, W_t converges to an equilibrium distribution. What is more, when ε is small, the equilibrium is approximately

$$W \sim \frac{1}{Z_T} e^{-\frac{1}{T}G(W)} \tag{9}$$

which you might recognize as a Gibbs distribution describing the equilibrium state of a system maintained at temperature T (in suitable units). Or, alternatively, an exponential family with sufficient statistic $G(W)$ and natural parameter $-1/T$. The point is that the equilibrium distribution favors small values of G, as

desired. What is more, the lower the "temperature" (i.e. the smaller the value of T), the more concentrated the distribution is at small values of $G(W)$.

This might look like a good start, but usually it isn't. The problem is that the trajectory, W_t, might spend enormous amounts of time stuck in the neighborhoods of local minima, even though these neighborhoods, collectively, have small probability mass under the equilibrium distribution (9). In other words, the convergence to (9) can be very slow.

In any case, the training of neural networks is more complicated because the noise usually comes from so-called batch training. Let $B_1, B_2, \ldots$ be a sequence of (random or systematically chosen) subsets of the indices of the training set, $B_k \in \{1, 2, \ldots, n\}$. In batch training,

$$\hat{\nabla}_W G(W_t) = \frac{1}{|B_t|} \sum_{k \in B_t} \nabla_W g_k(W_t)$$

If B_t is random in the sense that every $k \in \{1, 2, \ldots, n\}$ is equally likely to be chosen, then $\hat{\nabla}_W G(W_t)$ is indeed unbiased, but certainly not independent from one t to another; eventually the batches will overlap. Convergence theorems are harder to come by, but the observed benefits are not so different from those enjoyed by simple additive noise, at least with regards to an improved exploration of the loss landscape.

Stochastic gradient descent with slowing learning rate. In a related and widely used approach the "learning rate" is made time dependent, in such a way that $\varepsilon = \varepsilon_t$ converges towards zero as $t \to \infty$. There are many variations, including methods that automatically adjust the rate during learning procedure.

Going back to the representation in (8), let us suppose that the learning rate is chosen to be an *a priori* fixed function of the iteration, t ($\varepsilon \to \varepsilon_t$):

$$W_t = W_{t-1} - \varepsilon_t \nabla_W G(W_{t-1}) + \sqrt{2\varepsilon_t T}\eta_{t-1}$$

Various asymptotic results come down to how fast $\varepsilon_t \to 0$. Assuming certain regularity and boundedness conditions:

(a) If $\varepsilon_t \to 0$ sufficiently slowly, then the limiting distribution on W_t will be exactly (as opposed to approximately) the Gibbs distribution in (9). But trajectory W_t, itself, is guaranteed to continue to explore the entire surface. Forever.

(b) If $\varepsilon_t \to 0$ too rapidly, then the trajectory will converge to a limit, but not necessarily at a singularity. (Kind of like being stuck on the side of a mountain.)

Stochastic gradient descent with diminishing noise. Why not keep the learning rate constant (which, at least, would eliminate the problem of getting stuck somewhere other than at a singular point), and instead decrease the noise amplitude (or what is the same thing, decrease the temperature)? In other words, replace T by $T_t \to 0$ and keep ε constant:

$$W_t = W_{t-1} - \varepsilon \nabla_W G(W_{t-1}) + \sqrt{2\varepsilon T_t}\eta_t$$

This is an example of "simulated annealing."

Of course one could also try lowering the learning rate *and* the temperature. This might or might not be the ticket to great improvements, but a definitive solution is quite complicated and problem dependent.

In fact, you can try for yourself, in the next homework...

III Graphical Models

Goal: Develop tools for designing & computing with high-dimensional distributions

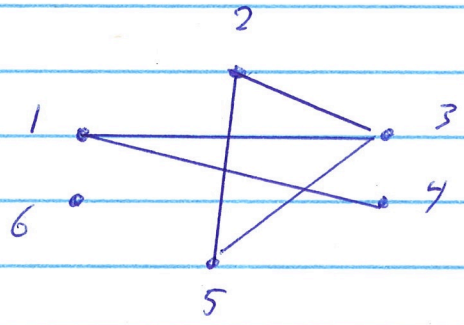
Main tool: Graphs for managing dependency among large numbers of RVs

Applications: image processing, signal processing, Modelling complex systems, ...

A. Markov random fields & Gibbs random fields

~~Markov random fields & Gibbs Random distributions~~

graphs & cliques (notation is half the battle!)



$G = (V, E)$ "graph"

vertices

edges (undirected)

$V = \{1, 2, 3, 4, 5, 6\}$

$E = \{\{1,3\}, \{1,4\}, \{2,3\}, \{2,5\}, \{3,5\}\}$ ← no "self edges"

Defn $C \subseteq V$ is a **clique** if it is fully connected, i.e.

$$i, j \in C \Rightarrow \{i, j\} \in E$$

A clique C is **maximal** if is not a (proper) subset of any other clique

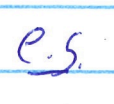
e.g. $\{6\}, \{1,3\}, \{2,3,5\}$ maximal cliques
$\{5\}, \{3,5\}$ cliques, but not maximal
$\{1,3,4\}$ not a clique

Notation $\mathcal{C}(G)$: set of all cliques in $G=(V,E)$

e.g. 12 cliques in above example

$$\mathcal{C}(G) = \{\underbrace{\{1\},\{2\},\ldots,\{6\}}_{\text{singletons}}, \underbrace{\{1,3\},\{1,4\},\{2,3\},\{2,5\},\{3,5\}}_{\text{edges}}, \{2,3,5\}\}$$

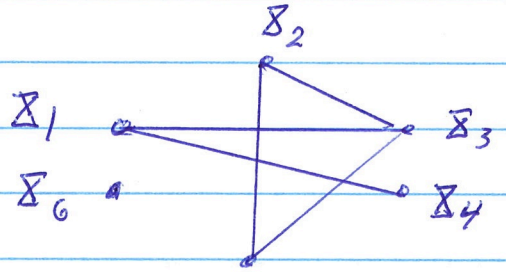
$G=(V,E)$ is a "complete graph" if $\forall i,j \in V$, $i \neq j$, $\{i,j\} \in E$

e.g.  (7 cliques)

(in general, $\mathcal{C}(G)$ complete $\Rightarrow |\mathcal{C}(G)| = 2^{|V|} - 1$)

Gibbs random fields

We will use graphs to help build distributions (models), and to help capture the dependency structure among the RV's



One random variable, X_v, associated with every vertex (aka site) $v \in V$

Each RV has a state space $X_v \in \Omega_v$

e.g. $\Omega_1, \Omega_2, \Omega_3 = \{1,2\}$
$\Omega_4, \Omega_5, \Omega_6 = \{1,2,3\}$

$$\underline{X} = (X_1, \ldots, X_6) = X_{1:6} \in \Omega = \Omega_1 \times \Omega_2 \times \cdots \times \Omega_6$$

Similarly, for any $A \subseteq V$, $A = \{v_1, \ldots, v_m\}$

$$X_A = (X_{v_1}, X_{v_2}, \ldots, X_{v_m}) \in \prod_{i=1}^{m} \Omega_{v_i} = \Omega_A$$

e.g. $A = (2,5)$ $X_A = (X_2, X_5) \in \Omega_2 \times \Omega_5$

$$= \{(1,1), (1,2), (1,3), (2,1), (2,2), (2,3)\}$$

$$= \Omega_A$$

Defn X is a GRF (Gibbs random field) wrt G if

$$p_X(x) = \frac{1}{Z} \prod_{C \in \mathcal{C}(G)} \phi_C(x_C) \quad \forall\ x \in \Omega$$

for some functions $\phi_C : \Omega_C \to [0, \infty)$ $\forall\ C \in \mathcal{C}(G)$ & some positive constant Z

Remarks: 1. Use f_X instead of p_X when X cont.

2. Z is called the "partition function" and $\{\phi_C\}$ are called the "clique functions"

3. GRF's are typically defined by requiring strict positivity:

$$\phi_C : \Omega_C \to (0, \infty) \text{ (instead of } [0, \infty))$$

e.g. G as in earlier example,

$$p_X(x) = \frac{1}{Z} x_1^{x_3} (x_1 + x_4) \left|\cos(x_2 \cdot x_3 \cdot x_5)\right| \left|\tan(x_2 + x_3)\right| x_6$$

$\phi_{(1,3)}$ $\phi_{(1,4)}$ $\phi_{(2,3,5)}$ $\phi_{(2,3)}$ ϕ_6

e.g. $\frac{1}{Z} e^{x_1 x_2 + x_2^{x_3} + \log(x_3 + x_4)}$ → "potential energies"

$= \frac{1}{Z} e^{x_1 x_2} e^{x_2^{x_3}} (x_3 + x_4)$

Clique functions

G: $x_1 - x_2 - x_3 - x_4$

"nearest neighbor interactions"

More remarks:

1. Representation is not unique (e.g. can always write using only maximal cliques - e.g. write

$$\psi_{(2,3,5)}(x_2, x_3, x_5) = |\cos(x_2 \cdot x_3 \cdot x_5)| |\tan(x_2 + x_3)|$$

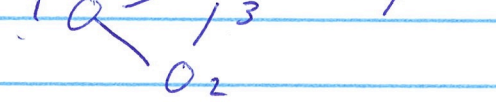
$$\& \quad \phi_{(2,3,5)} \quad \phi_{(3,3)}$$

instead of $\phi_{(2,3,5)}(x_2, x_3, x_5) \phi_{(2,3)}(x_2, x_3)$)

2. Every dist is GRF wrt the complete graph

3. We say that p_X "respects G" if X is GRF wrt G, e.g.

~~All GRF~~ $p_X(x) = \frac{1}{Z}(x_1 + x_2 + x_3) x_2^{x_3} |\log(x_3 + x_4)|$ respects



4. X GRF wrt $G = (V, E) \Rightarrow X$ GRF wrt $G' = (V, E')$ for any E' s.t. $E \subseteq E'$

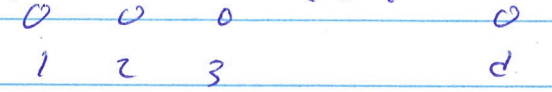
(Since $\mathcal{C}(G) \subseteq \mathcal{C}(G')$ & can just take $\phi_C = 1$ for any

$$C \in \mathcal{C}(G') \setminus \mathcal{C}(G)$$

More examples

e.g. $X_{1:d}$ ind $\Rightarrow$ $f_{X_{1:d}}(x_1, \dots, x_d) = \prod_1^d f_{X_k}(x_k)$

respects (is GRF wrt)



no edges

e.g $X_{1:d} \sim$ iid Bern$(\frac{1}{3})$ (so $X \in \{0,1\}$, $\mathbb{P}(X=1) = \frac{1}{3}$)

$$p_X(x) = \prod_{k=1}^d \left(\frac{1}{3}\right)^{x_k} \left(\frac{2}{3}\right)^{1-x_k}$$

e.g (Markov Chain) $X = (X_1, \dots, X_d)$

$$p_X(x) = \mathbb{P}(X_1 = x_1)\,\mathbb{P}(X_2 = x_2 \mid X_1 = x_1)\,\mathbb{P}(X_3 = x_3 \mid X_1 = x_1, X_2 = x_2)$$
$$\cdots \mathbb{P}(X_d = x_d \mid X_1 = x_1, X_2 = x_2, \dots, X_{d-1} = x_{d-1})$$

this is always true

$$= \mathbb{P}(X_1 = x_1)\,\mathbb{P}(X_2 = x_2 \mid X_1 = x_1)\,\mathbb{P}(X_3 = x_2 \mid X_2 = x_2) \cdots \mathbb{P}(X_d = x_d) \mid X_{d-1} = x_{d-1})$$

ϕ_1 $\quad$ $\phi_{(1,2)}$ $\quad$ $\phi(x_3, x_4)$ $\quad$ $\phi(x_{d-1}, x_d)$

$$= \phi_1(x_1) \prod_{k=2}^d \phi_{(k-1,k)}(x_{k-1}, x_k)$$

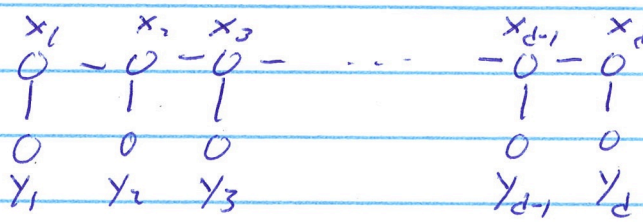
respects

linear, nearest-nbr graph

e.g. (hidden Markov model - "HMM")

$X = (X_1, \ldots, X_d)$ ~ Markov Chain
$Y = (Y_1, \ldots, Y_d)$
} $2d$ RV's

$$P(X = x, Y = y) = \phi_1(x_1) \prod_{k=2}^{d} \phi_{k-1,k}(x_{k-1}, x_k) \prod_{k=1}^{d} \psi_k(x_k, y_k)$$

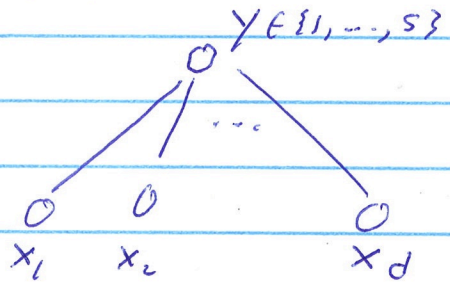


e.g. Naive Bayes $X \in \mathbb{R}^d$, $Y \in \{1, 2, \ldots, s\}$

$P(X = x, Y = y)$ Joint dist $\pi_y f_y(x)$

$$= \pi_y \prod_{i=1}^{d} f_{y,i}(x_i)$$

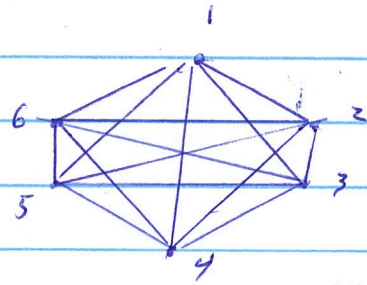
$\pi_y \to \phi(y)$ $\quad f_{y,i} \to \phi(y, x_i)$



e.g. multinomial
Roll fair die 5 times, let X_k = # times k appears
$\Omega_k = \{0, 1, \ldots, 5\}$

$$p_{X_{1:6}}(x_{1:6}) = \frac{5!}{x_1! x_2! \cdots x_6!} \left(\frac{1}{6}\right)^5 = \frac{5!}{6^5} \prod_{1}^{6} \frac{1}{x_k!} \prod_{\sum_1^6 x_k = 5}$$

$\Rightarrow$ Complete graph!



Independence, conditioning, marginalizing

What does the graph structure tell us about the distribution & the dependencies among the RV's?

In general, and very loosely, think of dependence as "flowing" through the links.

Here we develop some general rules about the presence & absence of links. Much of this is key to efficient computation & estimation in GRF's

Independence Given $\underline{X}_A = \{\underline{X}_k\}_{k \in A}$, $\underline{X}_B = \{\underline{X}_\ell\}_{\ell \in B}$ where $A \cap B = \phi$

$\underline{X}_A$ & $\underline{X}_B$ are independent if

$$p_{\underline{X}_A, \underline{X}_B}(x_A, x_B) = p_{\underline{X}_A}(x_A)\, p_{\underline{X}_B}(x_B)$$

Now suppose we have, instead,

$$p_{\underline{X}_A, \underline{X}_B}(x_A, x_B) = \frac{1}{Z}\, \phi(x_A)\, \psi(x_B)$$

Are $\underline{X}_A$ & $\underline{X}_B$ necessarily independent?

Yes: $p(x_A, x_B) = \frac{1}{Z}\phi_A(x_A)\phi_B(x_B)$

$\Rightarrow p(x_A) = \sum_{x_B} p(x_A, x_B) = \frac{1}{Z}\phi_A(x_A)\sum_{x_B}\phi_B(x_B) \propto \phi_A(x_A)$

$\propto$ proportional

Similarly, $p(x_B) \propto \phi_B(x_B)$

$\Rightarrow p(x_A, x_B) = \frac{1}{Z}\phi_A(x_A)\phi_B(x_B) = \frac{1}{Z'}p(x_A)p(x_B)$

$\Rightarrow Z' = 1 \Rightarrow p(x_A, x_B) = p(x_A)p(x_B)$ i.e. $\mathbf{x}_A \perp\!\!\!\perp \mathbf{x}_B$

$\perp\!\!\!\perp$ independent

Which we can use to prove

<u>Thm</u> $\mathbf{x}$ GRF wrt $G = (V, E)$ and if $A \cup B = V$, $A \cap B \neq \emptyset$ and $A \setminus B$ are <u>disconnected</u> in G (no path from anywhere in A to anywhere in B, & $A \cap B = \emptyset$) then $\mathbf{x}_A \perp\!\!\!\perp \mathbf{x}_B$

<u>Pf</u> no path $\Rightarrow \forall C \in \mathcal{C}(G)$, either $C \subseteq A$ or $C \subseteq B$

$$\Rightarrow p_{\mathbf{x}_A, \mathbf{x}_B}(x_A, x_B) = \prod_{\substack{C \subseteq A \\ C \in \mathcal{C}(G)}} \phi_C(x_C) \prod_{\substack{C \subseteq B \\ C \in \mathcal{C}(G)}} \phi_C(x_C)$$

$\Rightarrow \mathbf{x}_A \perp\!\!\!\perp \mathbf{x}_B$

Which we can generalize to

Corollary If $A, B \subseteq V$ are disconnected in G, then $X_A \perp\!\!\!\perp X_B$

Pf Let $\tilde{A} \triangleq \{v \in V : \exists \text{ path from } v \text{ to } A\}$
$\tilde{B} \triangleq \{v \in V : \exists \text{ path from } v \text{ to } B\}$
$\tilde{C} \triangleq V \setminus (\tilde{A} \cup \tilde{B}) \quad (= (\tilde{A} \cup \tilde{B})^c)$

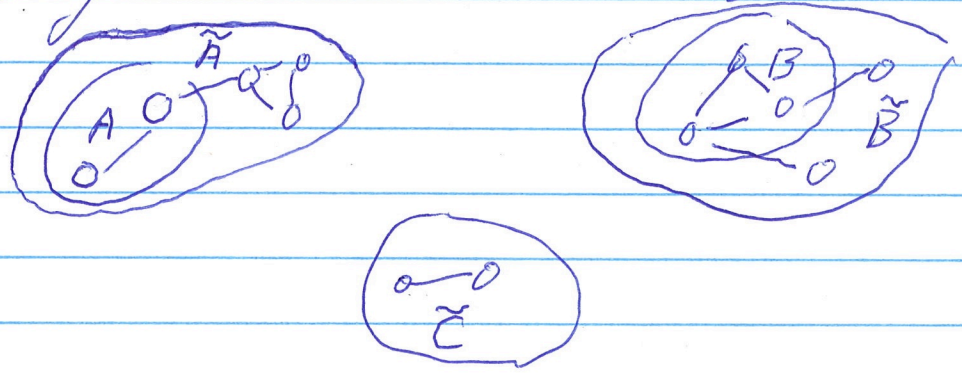
Then, by construction, every clique $c \in \mathcal{C}(G)$ is in exactly one of $\tilde{A}, \tilde{B}$, or $\tilde{C}$. Hence

$$p_X(x) = p_{X_{\tilde{A}}, X_{\tilde{B}}, X_{\tilde{C}}}(x_{\tilde{A}}, x_{\tilde{B}}, x_{\tilde{C}})$$

$$= \prod_{\substack{c \subseteq \tilde{A} \\ c \in \mathcal{C}(G)}} \phi_c(x_c) \prod_{\substack{c \subseteq \tilde{B} \\ c \in \mathcal{C}(G)}} \phi_c(x_c) \prod_{\substack{c \subseteq \tilde{C} \\ c \in \mathcal{C}(G)}} \phi_c(x_c)$$

$$\Rightarrow X_{\tilde{A}} \perp\!\!\!\perp X_{\tilde{B}} \perp\!\!\!\perp X_{\tilde{C}}$$

(last step by generalizing earlier argument about dist's that factor)



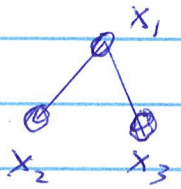
Remarks:

1. Does lack of edge imply independence? no

e.g. $X_1 \sim \text{Bern}(\frac{1}{2})$

if $X_1 = 0$, $X_2, X_3 \sim \text{Bern}(0.01)$
if $X_1 = 1$, $X_2, X_3 \sim \text{Bern}(0.99)$

$\Rightarrow p(x_1, x_2, x_3) = p(x_1) p(x_2 | x_1) p(x_3 | x_1)$

$\Rightarrow$ 

X_2, X_3 clearly not ind

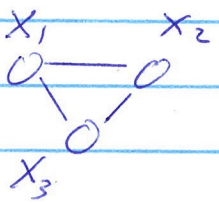
(though they are conditionally ind, given X_1)

2. Does edge imply dependence? no

$X_1, X_2 \sim$ iid $\text{Bern}(\frac{1}{2})$

given $X_1 = x_1$ $X_2 = x_2$ $X_3 \sim \text{Bern}\left(\frac{1}{2 + x_1 x_2}\right)$

$p(x_1, x_2, x_3) = p(x_1) p(x_2) p(x_3 | x_1, x_2)$

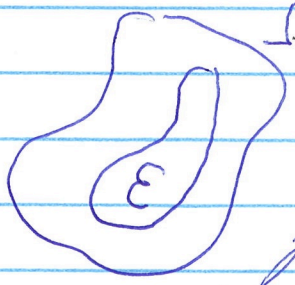
$\Rightarrow$ 

but $X_1 \perp\!\!\!\perp X_2$

Conditioning

What graph does $\mathbb{X}_A \mid \mathbb{X}_{A^c} = x_{A^c}$ respect?

in general



$E \subseteq \Omega$

$p_{\mathbb{X}}(x \mid \mathbb{X} \in E)$ same dist (ratios same) but renormalized

$$p_{\mathbb{X}_A}(x_A \mid \mathbb{X}_{A^c} = x_{A^c}) = \frac{p_{\mathbb{X}}(x_A, x_{A^c})}{p_{\mathbb{X}_{A^c}}(x_{A^c})}$$

← same graph, but with $\mathbb{X}_v$, $v \in A^c$, fixed ("clamped") at $\mathbb{X}_v = x_v$

↑ constant

Thm $\mathbb{X} \sim$ GRF wrt $G = (V, E)$, $A \subseteq V \Rightarrow$ for any x_{A^c}

$$p_{\mathbb{X}_A}(x_A \mid \mathbb{X}_{A^c} = x_{A^c})$$

respects $G' = (A, E')$, where

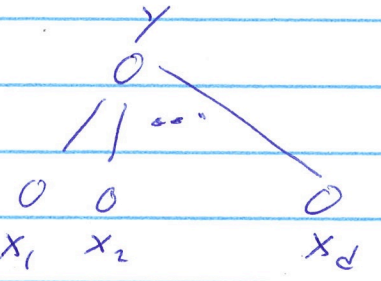
$$E' = \{\{i,j\} : i,j \in A, \{i,j\} \in E\}$$

PF See HW

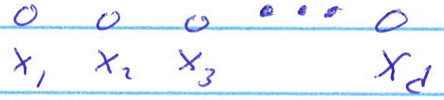
e.g. Naive Bayes

$$p_{X,Y}(x,y) = \pi_y \prod_{i=1}^{d} f_{y,i}(x_i)$$

$\phi(y)$ $\quad$ $\phi(y, x_i)$



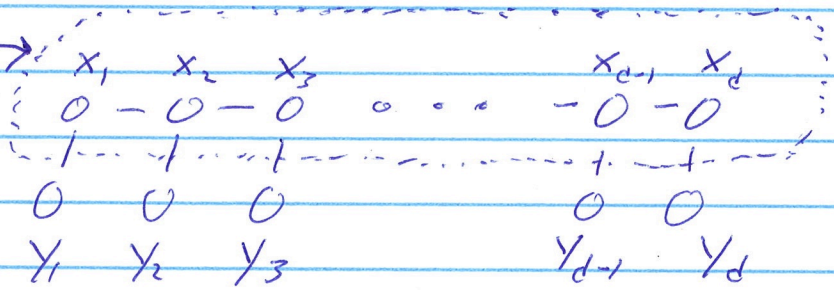
$$p(x \mid Y = c) = \prod_{i}^{d} f_{c,i}(x_i)$$



Conditionally independent

e.g. HMM $p(x,y) = \phi_1(x_1) \prod_{k=2}^{d} \phi_{k-1,k}(x_{k-1}, x_k) \prod_{\ell=1}^{d} \psi_\ell(x_\ell, y_\ell)$

$p(x \mid Y = y)$: →



$p(x \mid Y = y)$ has same graph structure as $p(x)$ (the original Markov chain)

Marginalizing

If W & Z are RV's with ~~joint~~ joint distribution $p_{W,Z}(w,z)$, then

$$p_W(w) = \sum_z p_{W,Z}(w,z)$$

is called the "marginal" dist. on W.

$X \sim$ GRF on $G=(V,E)$, $A \subseteq V$, what graph does

$$p_{X_A}(x_A) \quad \left(= \sum_{x_{A^c}} p_X(x)\right)$$

respect?

Thm $X \sim$ GRF wrt $G=(V,E)$, $A \subseteq V \Rightarrow$ $X_A \sim$ GRF wrt $G''(A,E'')$, where

$E'' = \{\{i,j\} : i,j \in A$ and either

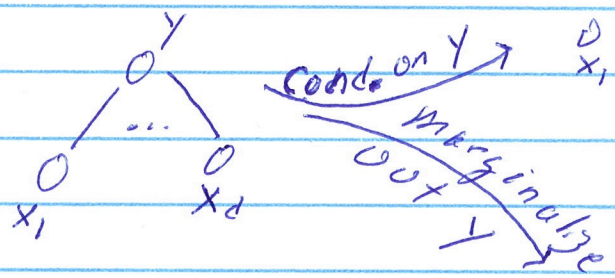
(i) $\{i,j\} \in E$, or

(ii) $\exists$ path from i to j that runs only through A^c $\}$

Pf See HW

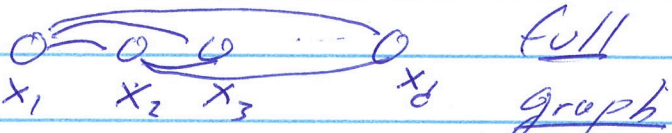
e.g. Naive Bayes

$$p_{\theta,\Sigma}(x,c) = \pi_c \prod_{k=1}^{d} f_{c,k}(x_k)$$



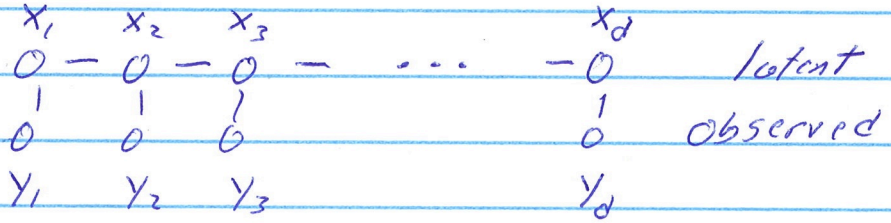
x_1 x_2 ... x_{d-1} x_d ind

(path from x_i to x_j thru y $\forall i,j$)



full graph

e.g. HMM



Goal: Use y to get x

Marginal ~~to~~ y: complete graph $\Rightarrow$ rich model for observations

Condition on y: x respects some graph (linear, nearest nbr) $\Rightarrow$ recursive alg's for recovering x from y (e.g. Kalman Filter, Viterbi algorithm, Forward-Backward algorithm ...)

Summary of the dependency rules in Gibbs random fields

- (**Independence**) If X is a GRF wrt $G = (V, E)$, if $A \subseteq V$, and $B \subseteq V$, and if there is no path from any $v \in A$ to any $u \in B$, then X_A and X_B are independent.
- (**Conditioning**) Assume that X is a GRF wrt $G = (V, E)$. Let $A \subseteq V$ and let $A^c = V \setminus A$. Then for any $x_{A^c} \in \Omega_{A^c}$, the conditional probability of X_A given $X_{A^c} = x_{A^c}$ is a GRF wrt $G' = (A, E')$, where E' is the set of edges among vertices in A in the original graph G.
- (**Marginalizing**) Assume that X is a GRF wrt $G = (V, E)$. Let $A \subseteq V$ and let $A^c = V \setminus A$. Then X_A is a GRF wrt $G'' = (A, E'')$, where $\{i, j\} \in E''$ if either $i, j \in A$ and $\{i, j\}$ was in the original graph, G, or $i, j \in A$ and there is a path from i to j in G that runs only through A^c.

Remarks:

1. To go from a joint distribution on several variables (e.g. $p_{X_A,X_B}(x_A, x_B)$) to a joint distribution on a subset of those variables (e.g. $p_{X_A}(x_A)$) you need to sum the joint pmf over all possible values of the variables that are not in the subset (e.g. $p_{X_A}(x_A) = \sum_{x_B} p_{X_A,X_B}(x_A, x_B)$). This procedure is sometimes referred to as "marginalizing," and the resulting pmf on the selected subset is sometimes referred to as the "marginal distribution on the subset" (e.g. $p_{X_A}(x_A)$).
2. Keep in mind that the conditioning and marginalizing rules are "worst case" — in special cases some of the resulting edges can be eliminated. But in general, all of the edges will be needed.
3. The random variables in a set of random variables, X_A, are independent if and only if their joint pmf factors into the product of their individual pmf's: $p_{X_A}(x_A) = \prod_{v \in A} p_{X_v}(x_v)$. Equivalently, the joint pmf factors into a product of functions of the individual components: $p_{X_A}(x_A) = \prod_{v \in A} \phi_v(x_v)$.
4. Recall that we say that "X respects G" if X is GRF wrt G.
5. A graph $G = (V, E)$ is *minimal* for X if X respects $G(V, E)$ but X does not respect any $G'(V, E')$, where $E' \subset E$ but $E' \neq E$.
6. *Having an edge* $\{u, v\}$ in a graph does not imply that X_u and X_v are dependent, and *not having an edge* $\{u, v\}$ does not imply that they are independent.

Markov random fields (and the MRF/GRF equivalence)

There is another way to look at Gibbs random fields that can be quite useful.

Markov random fields are essentially (see below) the same as GRF's. So why bother? Because the MRF point of view is very different & can suggest

* Computational algorithms for sampling & estimation (e.g. "belief propagation" & "Gibbs sampler")
* Different approaches to parameter estimation
* Different approaches to modelling

Notation: $G = (V, E)$.

$\forall i \in V$, $\mathcal{N}_i \triangleq \{j \in V : \{i,j\} \in E\}$ $\{\mathcal{N}_i\}_{i \in V}$ "neighborhood system"

Write $_i X$ to mean X_{i^c}, i.e.

$X_{V \setminus i}$ (all vertices except i)

Defn Given a graph $G = (V, E)$, let $\underline{X}$ denote a collection of R.V.s, $\{X_i\}_{i \in V}$, one for each site $i \in V$

$\underline{X}$ is a Markov random field (MRF) w.r.t. G if

$$P(X_i = x_i \mid X_j = x_j) = P(X_i = x_i \mid X_{n_i} = x_{n_i})$$

whenever $P(X_j = x_j) > 0$

Thm (Hammersly Clifford) If $p(x) > 0$ $\forall x \in \Omega$, then

$$\underline{X} \text{ GRF wrt } G \iff \underline{X} \text{ MRF wrt } G$$

Remark: One direction (Gibbs $\Rightarrow$ MRF) is pretty easy. The other isn't.

There are many corollaries & consequences.

e.g. If $p \sim$ MRF, with $p(x) > 0$ $\forall x$, and if $A \subseteq V$, & $B = V \setminus (A \cup n_A)$

$\Rightarrow p$ Gibbs

$\Downarrow$

$$X_A \perp\!\!\!\perp X_B \mid X_{n_A}$$

e.g. 2. $p(x) > 0 \ \forall x \in \Omega$, Then For Markov Chains,

"One-sided Markov property" $\Leftrightarrow$ "Two-sided Markov property"

$$P(X_k = x_k \mid X_{1:k-1} = x_{1:k-1})$$

$$= P(X_k = x_k \mid X_{k-1} = x_{k-1})$$

$\forall k = 2:d$

$$P(X_k = x_k \mid X_{1:k-1} = x_{1:k-1}, X_{k+1:d} = x_{k+1:d})$$

$$= P(X_k = x_k \mid X_{k-1} = x_{k-1}, X_{k+1} = x_{k+1})$$

$\forall k = 1:d$

(See HW)

III Graphical models

B Computing

One reason for organizing a probability distribution in terms of its implied graphical structure is that the graph can suggest feasible algorithms for a variety of computational tasks, including the computation of the normalizing constant, producing samples from the distribution, and computing estimates of its parameters.

We will look at two prominent examples: dynamic programming, which is exact but not always feasible, and Monte Carlo Markov Chain, which is only asymptotically exact, but often sufficient for a task when dynamic programming is too computationally intensive.

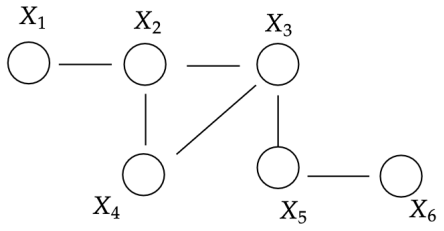
1 Dynamic programming

There are many applications to random fields, including finding the most likely state ($\text{argmax}_x\, p(x)$), computing expectations (e.g. of sufficient statistics), computing the normalizing constant (Z), and sampling (choosing $X \sim p$). We will discuss the last two, which illustrate all of the essential features of the DP approach to computation.

The notation can easily get in the way of understanding the simple concepts behind dynamic programming. To mitigate this, we will work with a particular graph, and then comment on the more-or-less straightforward generalization to arbitrary graphs.[1] For this purpose we will consider a probability distribution of the form

$$p(x) = \frac{1}{Z}\phi_1(x_1)\phi_{12}(x_1, x_2)\phi_{234}(x_2, x_3, x_4)\phi_{35}(x_3, x_5)\phi_{56}(x_5, x_6)$$

which respects the (minimal) graph



where each clique function, ϕ, is known. For simplicity, let us assume that each variable has the same range (set of possible values), Ω.

1. Compute Z
2. The boundary trick
3. Sampling from p

[1] Dynamic programming is a very broad term. In general, it applies to a kind of recursive "divide and conquer" approach, often related to graphical structures. For a much more complete description of dynamic programming for computing with Gibbs random fields, see the posted notes, "DP in a Nutshell."

Compute Z. Start with the definition

$$Z \doteq \sum_{x_1\in\Omega}\sum_{x_2\in\Omega}\sum_{x_3\in\Omega}\sum_{x_4\in\Omega}\sum_{x_5\in\Omega}\sum_{x_6\in\Omega} \phi_1(x_1)\phi_{12}(x_1,x_2)\phi_{234}(x_2,x_3,x_4)\phi_{35}(x_3,x_5)\phi_{56}(x_5,x_6) \tag{1}$$

which we can do by simply looping over all of the $|\Omega|^6$ possible values for the vector $x = (x_1, x_2, x_3, x_4, x_5, x_6)$. Alternatively, we can do this more efficiently with some careful bookkeeping.

Here are the steps:

1. Choose a *site visitation schedule*, meaning the order in which we will sum over the different components of x. Suppose for example that we choose $(1, 2, 3, 4, 5, 6)$.

2. Rewrite (1) to reflect the order in which the summations will be performed:

$$Z \doteq \sum_{x_6\in\Omega}\sum_{x_5\in\Omega} \phi_{56}(x_5,x_6) \sum_{x_4\in\Omega}\sum_{x_3\in\Omega} \phi_{35}(x_3,x_5) \sum_{x_2\in\Omega} \phi_{234}(x_2,x_3,x_4) \sum_{x_1\in\Omega} \phi_1(x_1)\phi_{12}(x_1,x_2)$$

   Notice that for each summation we have pulled out any terms that do not depend on the variable being summed.

3. The bookkeeping part is to define and record some arrays:

$$\begin{aligned} Z &= \sum_{x_6\in\Omega}\sum_{x_5\in\Omega} \phi_{56}(x_5,x_6) \sum_{x_4\in\Omega}\sum_{x_3\in\Omega} \phi_{35}(x_3,x_5) \sum_{x_2\in\Omega} \phi_{234}(x_2,x_3,x_4) \sum_{x_1\in\Omega} \phi_1(x_1)\phi_{12}(x_1,x_2) \\ &= \sum_{x_6\in\Omega}\sum_{x_5\in\Omega} \phi_{56}(x_5,x_6) \sum_{x_4\in\Omega}\sum_{x_3\in\Omega} \phi_{35}(x_3,x_5) \sum_{x_2\in\Omega} \phi_{234}(x_2,x_3,x_4) \underbrace{\sum_{x_1\in\Omega} \phi_1(x_1)\phi_{12}(x_1,x_2)}_{T_1(x_2)} \\ &= \sum_{x_6\in\Omega}\sum_{x_5\in\Omega} \phi_{56}(x_5,x_6) \sum_{x_4\in\Omega}\sum_{x_3\in\Omega} \phi_{35}(x_3,x_5) \underbrace{\sum_{x_2\in\Omega} \phi_{234}(x_2,x_3,x_4) T_1(x_2)}_{T_2(x_3,x_4)} \\ &= \sum_{x_6\in\Omega}\sum_{x_5\in\Omega} \phi_{56}(x_5,x_6) \underbrace{\sum_{x_4\in\Omega}\sum_{x_3\in\Omega} \phi_{35}(x_3,x_5) T_2(x_3,x_4)}_{T_3(x_4,x_5)} \\ &= \sum_{x_6\in\Omega}\sum_{x_5\in\Omega} \phi_{56}(x_5,x_6) T_3(x_4,x_5) \\ &= \sum_{x_6\in\Omega} \underbrace{\sum_{x_5\in\Omega} \phi_{56}(x_5,x_6) T_3(x_4,x_5)}_{T_4(x_5)} \\ &= \underbrace{\sum_{x_6\in\Omega} T_4(x_5)}_{T_5(x_6)} \\ &= \underbrace{T_5(x_6)}_{T_6} \end{aligned}$$

4. As for computing Z, we're done: $Z = T_6$.

Remarks on computational cost.

(i) Let's add up the costs:

Term	Computation	loops	Cost
$T_1(x_2)$	for every $x_2 \in \Omega$ we summed over $x_1 \in \Omega$	double	$O(\|\Omega\|^2)$
$T_2(x_3, x_4)$	for every $x_3, x_4 \in \Omega$ we summed over $x_2 \in \Omega$	triple	$O(\|\Omega\|^3)$
$T_3(x_4, x_5)$	for every $x_4, x_5 \in \Omega$ we summed over $x_3 \in \Omega$	triple	$O(\|\Omega\|^3)$
$T_4(x_5)$	for every $x_5 \in \Omega$ we summed over $x_4 \in \Omega$	double	$O(\|\Omega\|^2)$
$T_5(x_6)$	for every $x_6 \in \Omega$ we summed over $x_5 \in \Omega$	double	$O(\|\Omega\|^2)$
T_6	we summed over $x_6 \in \Omega$	single	$O(\|\Omega\|)$

for a total of something like $2|\Omega|^3 + 3|\Omega|^2 + |\Omega|$.

(ii) Could we have done better with a better choice of the schedule? Try switching the order of the sums over x_3 and x_4 (i.e. use the schedule (1,2,4,3,5,6)):

$$\begin{aligned} Z &= \sum_{x_6\in\Omega}\sum_{x_5\in\Omega}\phi_{56}(x_5,x_6)\sum_{x_3\in\Omega}\phi_{35}(x_3,x_5)\sum_{x_4\in\Omega}\sum_{x_2\in\Omega}\phi_{234}(x_2,x_3,x_4)\sum_{x_1\in\Omega}\phi_1(x_1)\phi_{12}(x_1,x_2) \\ &= \sum_{x_6\in\Omega}\sum_{x_5\in\Omega}\phi_{56}(x_5,x_6)\sum_{x_3\in\Omega}\phi_{35}(x_3,x_5)\sum_{x_4\in\Omega}\sum_{x_2\in\Omega}\phi_{234}(x_2,x_3,x_4)\underbrace{\sum_{x_1\in\Omega}\phi_1(x_1)\phi_{12}(x_1,x_2)}_{T_1(x_2)} \end{aligned} \quad (2)$$

$$\underbrace{\sum_{x_4\in\Omega}\sum_{x_2\in\Omega}\phi_{234}(x_2,x_3,x_4)\,T_1(x_2)}_{T_2(x_3,x_4)}$$

$$\underbrace{\sum_{x_3\in\Omega}\phi_{35}(x_3,x_5)\,T_2(x_3,x_4)}_{T_4(x_3)}$$

$$\underbrace{\sum_{x_5\in\Omega}\phi_{56}(x_5,x_6)\,T_4(x_3)}_{T_3(x_5)}$$

$$\underbrace{\sum_{x_6\in\Omega}\phi_{56}(x_5,x_6)\,T_3(x_5)}_{T_5(x_6)}$$

$$\underbrace{\sum_{x_6\in\Omega}T_5(x_6)}_{T_6}$$

Notice that the switch allows us to avoid one of the two triple loops, thereby lowering the cost to approximately $|\Omega|^3 + 4|\omega|^2 + |\Omega|$ instead of $2|\Omega|^3 + 3|\Omega|^2 + |\Omega|$. If Ω were large, then this would end up being about half of the original cost.

Hang onto these six arrays, $T_1(x_2)$, $T_2(x_3, x_4)$, $T_4(x_3)$, $T_3(x_5)$, $T_5(x_6)$ and T_6. They'll come in handy if we want to *sample* (and not just normalize) p.

(iii) But what is the *best* order? It's not too hard to see that in the current example it is impossible to do better than (1,2,4,3,5,6)[2], though there are several ways to match it.

As for a general graph, this turns out to be an interesting question, but with a disappointing answer: the problem is "NP-hard." This means that it is very unlikely that any algorithm exists which will not grow exponentially with the size of the graph. (But mathematicians are not quite ready to say for sure. In fact they may never be, i.e. it might not be "decidable"!)

The boundary trick. (Optional, but really handy.)

Although the best site-visitation schedule is difficult to identify, there is a neat trick for easily getting an upper bound on the cost for any given schedule. What's more, it provides a way to visualize dynamic programming:

Imagine going from vertex to vertex in the order dictated by the visitation schedule, e.g. (1,2,4,3,5,6). Upon arriving at a vertex, it gets filled in, or "painted," or otherwise marked—see Figure (1), where black vertices have already been painted and the red vertex in each panel is the one being visited.

The cost of visiting a vertex is one more than the size of the boundary of the painted vertices (including the current vertex). What is more, for any $v \in V$, $T_v(x) = T_v(x_{\eta_A})$, i.e. the boundary of A corresponds to the set of arguments of T_v. From the sequence of graphs we can read off the dependencies and, consequently, the number of loops over Ω. For example, $T_1(x) = T_1(x_2)$, and hence two loops, and $T_3(x) = T_3(x_5)$, also two loops. But $T_2(x) = T_2(x_3, x_4)$.

We therefore have an upper bound on the computational cost of each summation, and hence an upper bound on the total cost.[3]

[2]In this sense: there will always be at least one triple loop, and given that there is only one triple loop, there will be nor more than one single loop.

[3]If the painted vertices become disconnected, then one gets a better bound by defining A to be the connected component

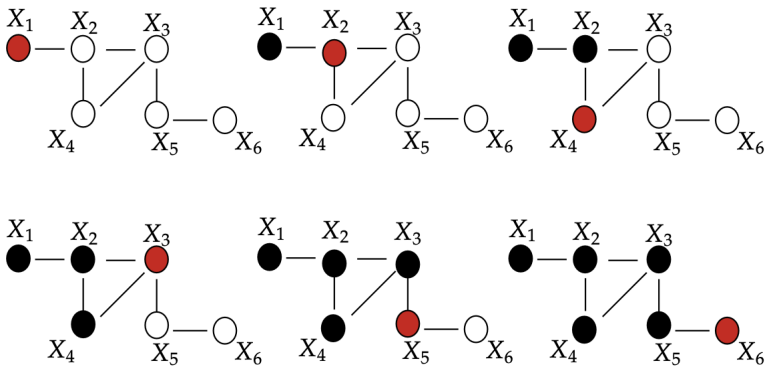


Figure 1: **Boundary Trick.** The visitation schedule, (1,2,4,3,5,6), dictates the order in which the vertices are painted, which corresponds to the order in which the variables are summed over. Black represents the vertices that have already been painted, and red represents the variable currently visited. Select one of the graphs (say the second, which correspond to visiting site 2), and let A be the set of unpainted (unfilled) vertices (e.g. in the case of the second graph, $A = \{3, 4, 5, 6\}$). Consistent with MRF notation, let η_A be the boundary of A, which is a subset of the unpainted vertices (e.g. $\eta_A = \{3, 4\}$ for the second graph). The number of loops required to compute T_2 is $b + 1$, where $b = |\eta_A|$ (e.g. three loops costing $|\Omega|^3$ in the case of the second graph).

Sampling from p. We can always rewrite p in terms of a sequence of conditional distributions. Actually there are many ways to do this, one for each ordering of the sites in V. Consider, for example, the identity

$$p(x) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_2, x_3, x_4)p(x_6|x_1, x_2, x_3, x_4, x_5))$$

or, in a different order, consider

$$p(x) = p(x_6)p(x_5|x_6)p(x_4|x_5, x_6)p(x_3|x_4, x_5, x_6)p(x_2|x_3, x_4, x_5, x_6)p(x_1|x_2, x_3, x_4, x_5, x_6) \tag{3}$$

Our goal here is to draw a sample, $X \sim p(x)$. This would be easy if we could compute each of the conditional probabilities in (3):

$$\text{select } X_6 \sim p(x_6),\ X_5 \sim p(x_5|X_6),\ X_3 \sim p(x_3|X_5, X_6)\ \cdots\ X_1 \sim p(x_1|X_2, X_3, X_4, X_5, X_6)$$

Each step could be executed using a standard random number generator. The process could be repeated many times to efficiently produce a sequences of (pseudo) random samples from p.

So how are we to compute these conditional probabilities? Since each conditional probability is a ratio of marginal probabilities (e.g. $p(x_3|X_5, X_6) = p(x_3, X_5, X_6)/p(x_5, x_6)$), the challenge comes down to computing the relevant marginal probabilities. Concerning the particular factorization in (3) we've already computed all of the necessary marginal probabilities! In particular, every necessary summation has already been made and stored the T arrays:

of painted vertices that includes the current (red) site.

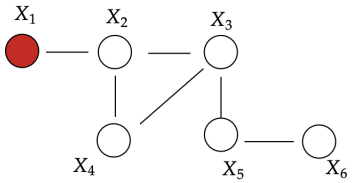
$$
\begin{aligned}
p(x_1, x_2, x_3, x_4, x_5, x_6) &= \frac{1}{Z}\phi_1(x_1)\phi_{12}(x_1, x_2)\phi_{234}(x_2, x_3, x_4)\phi_{35}(x_3, x_5)\phi_{56}(x_5, x_6) \\
p(x_2, x_3, x_4, x_5, x_6) &= \frac{1}{Z}\phi_{234}(x_2, x_3, x_4)\phi_{35}(x_3, x_5)\phi_{56}(x_5, x_6)T_1(x_2) \\
p(x_3, x_4, x_5, x_6) &= \frac{1}{Z}\phi_{35}(x_3, x_5)\phi_{56}(x_5, x_6)T_2(x_3, x_4) \\
p(x_3, x_5, x_6) &= \frac{1}{Z}\phi_{35}(x_3, x_5)\phi_{56}(x_5, x_6)T_4(x_3) \\
p(x_5, x_6) &= \frac{1}{Z}\phi_{56}(x_5, x_6)T_3(x_5) \\
p(x_6) &= \frac{1}{Z}T_5(x_6) \\
Z &= T_6
\end{aligned}
$$

See Figure (2) for a summary of the derivations.

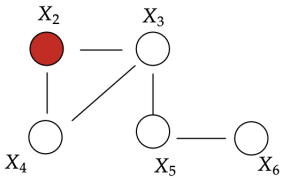
All that remains is to give explicit formulas for the conditional probabilities used in (3). Just plug in and cancel common terms:

$$
\begin{aligned}
p(x_6) &= \frac{1}{Z}\phi_{56}(x_5, x_6)T_5(x_6) = \frac{\phi_{56}(x_5, x_6)T_5(x_6)}{T_6} \\
p(x_5|x_6) &= \frac{p(x_5, x_6)}{p(x_6)} = \frac{T_3(x_5)}{T_5(x_6)} \\
p(x_3|x_5, x_6) &= \frac{p(x_3, x_5, x_6)}{p(x_5, x_6)} = \frac{\phi_{35}(x_3, x_5)T_4(x_3)}{T_3(x_5)} \\
p(x_4|x_3, x_5, x_6) &= \frac{p(x_4, x_3, x_5, x_6)}{p(x_3, x_5, x_6)} = \frac{T_2(x_3, x_4)}{T_4(x_3)} \\
p(x_2|x_3, x_4, x_5, x_6) &= \frac{p(x_2, x_3, x_4, x_5, x_6)}{p(x_3, x_4, x_5, x_6)} = \frac{\phi_{234}(x_2, x_3, x_4)T_1(x_2)}{T_2(x_3, x_4)} \\
p(x_1|x_2, x_3, x_4, x_5, x_6) &= \frac{p(x_1, x_2, x_3, x_4, x_5, x_6)}{p(x_2, x_3, x_4, x_5, x_6)} = \frac{\phi_1(x_1)\phi_{12}(x_1, x_2)}{T_1(x_2)}
\end{aligned}
$$

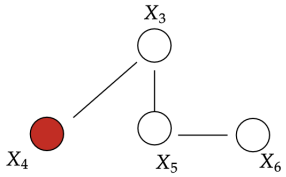
$$p(x_1, x_2, x_3, x_4, x_5, x_6) = \frac{1}{Z}\phi_1(x_1)\phi_{12}(x_1, x_2)\phi_{234}(x_2, x_3, x_4)\phi_{35}(x_3, x_5)\phi_{56}(x_5, x_6)$$



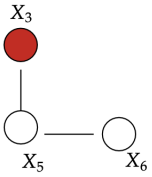
$$p(x_2, x_3, x_4, x_5, x_6) = \frac{1}{Z}\sum_{x_1 \in \Omega} \phi_1(x_1)\phi_{12}(x_1, x_2)\phi_{234}(x_2, x_3, x_4)\phi_{35}(x_3, x_5)\phi_{56}(x_5, x_6)$$
$$= \frac{1}{Z}\phi_{234}(x_2, x_3, x_4)\phi_{35}(x_3, x_5)\phi_{56}(x_5, x_6)T_1(x_2)$$



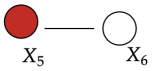
$$p(x_3, x_4, x_5, x_6) = \frac{1}{Z}\sum_{x_2 \in \Omega} \phi_{234}(x_2, x_3, x_4)\phi_{35}(x_3, x_5)\phi_{56}(x_5, x_6)T_1(x_2)$$
$$= \frac{1}{Z}\phi_{35}(x_3, x_5)\phi_{56}(x_5, x_6)T_2(x_3, x_4)$$



$$p(x_3, x_5, x_6) = \frac{1}{Z}\sum_{x_4 \in \Omega} \phi_{35}(x_3, x_5)\phi_{56}(x_5, x_6)T_2(x_3, x_4)$$
$$= \frac{1}{Z}\phi_{35}(x_3, x_5)\phi_{56}(x_5, x_6)T_4(x_3)$$



$$p(x_5, x_6) = \frac{1}{Z}\sum_{x_3 \in \Omega} \phi_{35}(x_3, x_5)\phi_{56}(x_5, x_6)T_4(x_3)$$
$$= \frac{1}{Z}\phi_{56}(x_5, x_6)T_3(x_5)$$



$$p(x_6) = \frac{1}{Z}\sum_{x_5 \in \Omega} \phi_{56}(x_5, x_6)T_3(x_5)$$
$$= \frac{1}{Z}T_5(x_6)$$

Figure 2: **Marginalization.** Beginning with the definition of $p(x)$, at the top, the figure depicts the graphs respected by each successive marginalization, in the order (1,2,4,3,5,6). The sites marked with red indicate which variable is being summed over, and the equation below each graph shows the resulting marginal probability. Since the T arrays were already computed during the calculation of Z, no further summations are necessary. Everything can be written in terms of the original clique functions and the stored previously computed arrays.

2 Gibbs Sampler: an introduction to Markov Chain Monte Carlo (MCMC)

1. Goals and approach
2. Thought experiment
3. Convergence from the information point of view
4. Prototype example: Ising model
5. Localization
6. Generalizations

Take a glance at Figure (3). We will discuss the Ising model shortly, but for now just consider any MRF (or GRF) that respects the lattice graph in the figure. Assume that the lattice is n by n. Can you find a site-visitation schedule that has maximum boundary less than n? In fact there is no way to paint the $n \times n$ lattice without at some point incurring a boundary of size n, at which point the dynamic-programming cost for a single variable will be $|\Omega|^n$—infeasible even for modest-sized graphs.

Goals and approach. We will use a slightly different notation: instead of using Ω to represent the state space of a single variable, $x_v,\ v \in V$, we will use it to represent the possible values (aka the "state space") for the entire vector $x = \{x_v\}_{v \in V} \in \Omega$.

Given a probability $p(x)$, the goal is to devise a way to choose a sequence $X(0), X(1), X(2), \ldots \sim p$ such that

(1) $\mathbb{P}\left(X(t) = x\right) \overset{t \longrightarrow \infty}{\longrightarrow} p(x)$

(2) For any function $g(x)$,

$$\frac{1}{T} \sum_{t=1}^{T} g(X(t)) \to \mathbb{E}_p[g(X)]$$

There are many applications, including parameter estimation for high-dimensional models, image and signal processing, estimating stochastic processes, and prediction.

The MCMC approach is to recursively select $X(t+1)$ from $X(t)$, which is to say, generate a *Markov chain*:

$$\mathbb{P}\left(X(t+1) = x \mid X(0) = x(0), X(1) = x(1), \ldots, X(t) = x(t)\right) = \mathbb{P}\left(X(t+1) = x \mid X(t) = x(t)\right)$$

for all $x \in \Omega$ and all $x(t) \in \Omega,\ t = 0 : T-1$. In words, a Markov chain is a sequence of random variables, indexed by time, for which the future given the present and the past is equal to the future given just the present. Each element of the sequence is called a state, which itself suggests the Markov property, namely that the state is sufficient for predicting the future.

Thought experiment. To get the idea, let us consider an extremely simple example: the target probability (i.e. the one from which we hope to generate samples) is a probability on just two random variables: $(X_1, X_2) \sim p(x_1, x_2) = \mathbb{P}(X_1 = x_1, X_2 = x_2),\ \forall x_1, x_2 \in \Omega$. Before going further, the following notation will prove convenient, especially when working with large numbers of random variables (instead of just two):

given $i \in \{1, 2\}$ define

$$ {}_ix = \begin{cases} x_1 & \text{if } i = 2 \\ x_2 & \text{if } i = 1 \end{cases} $$

Let π be a probability on $\{1, 2\}$ and consider the following recursion:

(a) initialize

1. $x_1(0), x_2(0) \in \Omega$, arbitrary
2. t=0

(b) iterate

1. choose a "site" to "visit": $i \sim \pi$
2. update $x_i(t)$: choose $x_i(t+1) \sim p(x_i \mid {}_iX(t) =_i x(t))$
3. do *not* update ${}_ix(t)$: ${}_ix(t+1) =_i x(t)$
4. set $t = t + 1$ and repeat

This is an example of the Gibbs sampler. Let $X(0), X(1), \ldots X(T)$ be the resulting (random) sequence. What happens to $X(t)$ as $t \to \infty$? What happens is that $X(t)$ satisfies each of the two goals, (1) and (2), articulated above.

We might think of the algorithm as an effort by $X_1(t)$ and $X_2(t)$ to come to an equilibrium, in which each is a selection from the conditional probability given the value of the other.

Convergence from the information point of view. One revealing way to prove convergence is to use information theory. Let q_t be the distribution on $(X_1(t), X_2(t))$. Then the result of an iteration of the sampler can be summarized, succinctly, in terms of $q_1, q_2, \ldots$ as follows

$$ q_{t+1}(x_1, x_2) = \pi_1 p(x_1 \mid x_2) q_t(x_2) + \pi_2 p(x_2 \mid x_1) q_t(x_1) \tag{4} $$

This gives us some intuition for why we might expect $q_t(x_1, x_2)$ to converge to $p(x_1, x_2)$: if site 1 is chosen, then

$$ \begin{aligned} q_t(x_1, x_2) &= q_t(x_1 \mid x_2) q_t(x_2) \\ &\to p_t(x_1 \mid x_2) q_t(x_2) \end{aligned} $$

which is, presumably, a step closer to $p_t(x_1, x_2)$. After all, $p_t(x_1, x_2) = p_t(x_1 \mid x_2) p_t(x_2)$, and it makes sense that $p_t(x_1 \mid x_2) q_t(x_2)$ is closer to $p_t(x_1 \mid x_2) p_t(x_2)$ than $q_t(x_1 \mid x_2) q_t(x_2)$. If site 2 is chosen, the argument is no different.

Using KL "distance," we can get pretty close to a proof out of this way of thinking: we will show that if $q_t \neq p$, then $D(q_{t+1} \| p) < D(q_t \| p)$. In other words, the distribution on $(X_1(t), X_2(t))$ gets closer to p with every iteration of the Gibbs sampler. (And if q_t is already equal to p, then you can use equation 4 to convince yourself that $q_{t+1} = p$ as well.)

Proposition. $D(q_{t+1}\|p) \leq D(q_t\|p)$

Proof. Fix t, and, with reference to equation (4), let

$$\begin{aligned} r_1(x_1, x_2) &= p(x_1 \mid x_2) q_t(x_2) \qquad \text{(the result of visiting } x_1 \text{ at } t+1) \\ r_2(x_1, x_2) &= p(x_2 \mid x_1) q_t(x_1) \qquad \text{(the result of visiting } x_2 \text{ at } t+1) \end{aligned}$$

Then

$$\begin{aligned} D(q_{t+1}\|p) &= D(\pi_1 r_1 + \pi_2 r_2 \| p) \qquad \text{(equation (4))} \\ &\leq \pi_1 D\left(r_1\|p\right) + \pi_2 D\left(r_2\|p\right) \qquad \text{(convexity of } D(f\|g) \text{ wrt } f) \end{aligned}$$

Now calculate $D(r_1\|p)$:

$$\begin{aligned} D(r_1\|p) &= D\left(p(X_1|X_2) q_t(X_2) \| p(X_1, X_2)\right) \\ &= \sum_{x_1}\sum_{x_2} p(x_1 \mid x_2) q_t(x_2) \log \frac{p(x_1 \mid x_2) q_t(x_2)}{p(x_1, x_2)} \\ &= \sum_{x_1}\sum_{x_2} p(x_1 \mid x_2) q_t(x_2) \log \frac{p(x_1 \mid x_2) q_t(x_2)}{p(x_1|x_2) p(x_2)} \\ &= \sum_{x_1}\sum_{x_2} p(x_1 \mid x_2) q_t(x_2) \log \frac{q_t(x_2)}{p(x_2)} \\ &= \sum_{x_2} q_t(x_2) \log \frac{q_t(x_2)}{p(x_2)} \\ &= D(q_t(X_2)\|p(X_2)) \end{aligned}$$

(This makes sense—we visited x_1 and changed $q_t(x_1|x_2)$ into $p(x_1|x_2)$. The only remaining difference between p and the updated q is the differences between $p(x_2)$ and $q_t(x_2)$.) As a consequence, I claim that $D(r_1\|p) \leq D(q_t\|p)$:

$$\begin{aligned} &D(r_1\|p) = D(q_t(X_2)\|p(X_2)) \\ &= \sum_{x_2} q_t(x_2) \log \frac{q_t(x_2)}{p(x_2)} \\ &= \sum_{x_2} q_t(x_2) \sum_{x_1} q_t(x_1 \mid x_2) \log \frac{q_t(x_2)}{p(x_2)} \\ &= \sum_{x_2} q_t(x_2) \left(\sum_{x_1} q_t(x_1 \mid x_2) \log \frac{q_t(x_2)}{p(x_2)} + \sum_{x_1} q_t(x_1 \mid x_2) \log \frac{q_t(x_1 \mid x_2)}{q_t(x_1 \mid x_2)} \right) \quad \text{(second term is zero)} \\ &\leq \sum_{x_2} q_t(x_2) \left(\sum_{x_1} q_t(x_1 \mid x_2) \log \frac{q_t(x_2)}{p(x_2)} + \sum_{x_1} q_t(x_1 \mid x_2) \log \frac{q_t(x_1 \mid x_2)}{p(x_1 \mid x_2)} \right) \quad (\forall x_2,\ D(q(X_1|x_2)\|p(X_1|x_2)) \geq 0) \\ &= \sum_{x_2} q_t(x_2) \left(\sum_{x_1} q_t(x_1 \mid x_2) \log \frac{q_t(x_2) q_t(x_1 \mid x_2)}{p(x_2) p(x_1 \mid x_2)} \right) \\ &= D(q_t\|p) \end{aligned}$$

With little more than copying, we also find that $D(r_2\|p) \leq D(q_t\|p)$. Finally, put this into the inequality $D(q_{t+1}\|p) \leq \pi_1 D\left(r_1\|p\right) + \pi_2 D\left(r_2\|p\right)$ and conclude that

$$D(q_{t+1}\|p) \leq D(q_t\|p)$$

Which proves the Proposition.

Remarks:

i. To go further, and conclude that $q_t \neq p \Rightarrow D(q_{t+1}\|p) < D(q_t\|p)$, we would need to examine each of the two inequalities that appeared in the proof. We will not go into detail, but it is not hard to verify that if $q_t \neq p$ then at least one of those inequalities must be a strict inequality.

ii. What is more, not only is $D(q_{t+1}\|p) < D(q_t\|p)$, but in fact $D(q_t\|p) \to 0$ as $t \to \infty$.

iii. For those of you who have had an earlier course on Markov chains, it is straightforward to show that the Markov sequence $X(0), X(1), \ldots$ is aperiodic with connected state space. The general theory then guarantees that both goals, (1) and (2), are achieved.

iv. **Most importantly, all of the arguments generalize immediately to sampling from a distribution with arbitrary number of variables, $X = (X_1, \ldots, X_d)$. In this situation, ${}_iX$ represents the variables $X_1, \ldots, X_d$ excepting X_i. The sampling algorithm is the obvious extension of the one we studied for $d = 2$: Given a distribution π on $\{1, 2, \ldots, d\}$, the description is identical:**

   (a) initialize
      1. $x_1(0), x_2(0), \ldots, x_d(0) \in \Omega$, arbitrary
      2. t=0
   (b) iterate
      1. choose a "site" to "visit": $i \sim \pi$
      2. update $x_i(t)$: choose $x_i(t+1) \sim p(x_i \mid {}_iX(t) = {}_ix(t))$
      3. do *not* update ${}_ix(t)$: ${}_ix(t+1) = {}_ix(t)$
      4. set $t = t + 1$ and repeat

v. If you look carefully at the proof of the proposition, it becomes apparent that random site visitation is not central to the convergence argument. To the contrary, *any* visitation schedule, random or otherwise, that guarantees that every site is visited infinitely often will also converge, and for essentially the same reason: $D(q_t\|p)$ is a monotonically decreasing sequence.

Prototype example: Ising model. The origins of MCMC are in physics, where many high-dimensional complex dynamical systems are analytically intractable but nevertheless lend themselves to simulation. The prototypical example is the Ising model for ferromagnetism:

Consider the $n \times n$ square lattice depicted in the figure below. Let V be the set of sites ($V = \{1, 2, \ldots, n\}^2$), and associate a random variable $X_s \in \{-1, 1\}$ with each site $s \in V$. Write X (resp. x) to represent the random vector $\{X_s\}_{s \in V}$ (resp. $\{x_s\}_{s \in V}$). Finally, for any pair of sites $s, t \in V$ we will write $< s, t >$ to indicate that s and t are horizontal or vertical neighboring pairs, as are u and v, as well as w and z, in Figure (3).

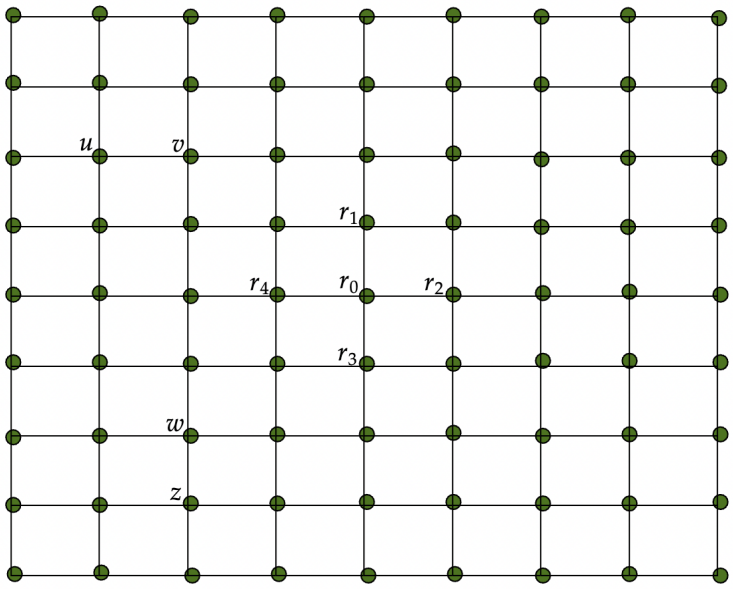


Figure 3: **Ising model.** V is the set of nodes, representing locations of single iron atoms in an idealized two-dimensional ferromagnet. Each atom $s \in V$ has a magnetic spin that is in one of two states, $X_s \in \{-1, 1\}$. Two sites are called neighbors if they are vertical (e.g. w and z or horizontal (e.g. u and v) nearest neighbors. Neighboring states interact. The notation $< s, t >$ indicates that $s, t \in V$ are neighboring sites.

The distribution on X is a function of the set of all pairs of random variables that are neighbors in the lattice:

$$p(x) = \frac{1}{Z_\beta} e^{\beta \sum_{<s,t>} x_s x_t} \tag{5}$$

where the summation, $\sum_{<s,t>}$, represents summation over all neighboring pairs $s, t \in V$, and β is a positive constant representing the inverse of the absolute temperature. Z_β is the normalizing constant, also known as the partition function.

The model is simple and intuitive. The most likely configurations are those for which neighboring pairs, say s and t, tend to line up: x_s and x_t are both 1 or both -1. If there are more spin states in one or the other of the two directions, then there is a macroscopic magnetic field. Ising had hoped to show that there would emerge a critical temperature in the large-lattice limit, signaling a phase transition to a stronger magnetic organization of the spins for temperatures below the critical temperature. But he was able to prove that no such temperature exists in the more elementary case of a one-dimensional lattice, and he assumed that the same conclusion was likely to hold in two and three-dimensional systems.

He was wrong. The Ising model in two dimensions turns out to be exactly what Ising was looking for: a relatively simple example of a system that exhibits phase transitions.

Sampling from the Ising model can reveal these transitions and other subtle aspects of the model. By far the most computationally expensive step in running the Gibbs sampler is step 2 in the iteration: Choose $x_i(t+1) \sim p(x_i \mid_i X(t) =_i x(t))$. How exactly is this done? Is it necessary to consider the spin states at every site in order to compute the conditional probability at just one site? Let's see what's involved.

Localization. Refer to Figure (3). Assume that $x(t) = \tilde{x}$ and that $r_0 \in V$ is the chosen site for the current update. The job at hand is to select a spin, $x_{r_0} \in \{-1, 1\}$, from the conditional probability on x_{r_0} given the spin states of all of the other variables, ${}_{r_0}\tilde{x}$. It will be convenient to define the *neighborhood*, $\mathcal{N}_{r_0}$, of r_0, to mean the set of nearest horizontal and vertical neighbors, $\mathcal{N}_{r_0} = \{r_1, r_2, r_3, r_4\}$. And, similarly, to define a larger neighborhood $\bar{\mathcal{N}}_{r_0}$ which includes r_0: $\bar{\mathcal{N}}_{r_0} = \{r_0, r_1, r_2, r_3, r_4\}$. With this notation:

$$
\begin{aligned}
p(x_{r_0}|_{r_0}\tilde{x}) &= \frac{p(x_{r_0},{}_{r_0}\tilde{x})}{p(x_{r_0}=-1,{}_{r_0}\tilde{x}) + p(x_{r_0}=1,{}_{r_0}\tilde{x})} \\
&= \frac{\frac{1}{Z_\beta} e^{\beta\left(x_{r_0}\sum_{i\in\mathcal{N}_{r_0}}\tilde{x}_{r_i} + \sum_{<s,t>:s,t\notin\bar{\mathcal{N}}_{r_0}}\tilde{x}_s\tilde{x}_t\right)}}{\frac{1}{Z_\beta} e^{\beta\left(-\sum_{i\in\mathcal{N}_{r_0}}\tilde{x}_{r_i} + \sum_{<s,t>:s,t\notin\bar{\mathcal{N}}_{r_0}}\tilde{x}_s\tilde{x}_t\right)} + \frac{1}{Z_\beta} e^{\beta\left(\sum_{i\in\mathcal{N}_{r_0}}\tilde{x}_{r_i} + \sum_{<s,t>:s,t\notin\bar{\mathcal{N}}_{r_0}}\tilde{x}_s\tilde{x}_t\right)}} \\
&= \frac{e^{\beta x_{r_0}\sum_{i\in\mathcal{N}_{r_0}}\tilde{x}_{r_i}}}{e^{-\beta\sum_{i\in\mathcal{N}_{r_0}}\tilde{x}_{r_i}} + e^{\beta\sum_{i\in\mathcal{N}_{r_0}}\tilde{x}_{r_i}}}
\end{aligned}
$$

The calculation is "local": the partition function Z_β disappears (fortunately!), and the conditional probability depends only on the nearest neighbors of the visited site r_0.

This last observation lends credence to the interpretation of this form of MCMC as simulating a physical system. Each site is influenced only by its neighbors. On occasion, its state changes (flips), most typically to the consensus state (if any) among its neighbors.

Generalizations.

i. **(block updating)** In the case of the Ising model, superimpose a checkerboard pattern and let $B \subset V$ be the black sites and $W \subset V$ be the white sites. Look again at the updating of the spin at a given site, say r_0, and recall that this only depends on the states at the neighboring sites, r_1, r_2, r_3, and r_4 in Figure (3). If, say, r_0 is black, then we might as well update *all* of the black sites simultaneously, since they are conditionally independent given the white sites, and then, similarly, update all of the white sites given the updated black sites. This opens the door to potentially far more efficient parallel computing. The general picture, for distributions defined on more general graph structures, is to first color the graph in such a way that no two neighbors have the same color, and then cycle through the colors updating simultaneously every site of the chosen color.

ii. **(simulated annealing)** If you heat up a physical system and then slowly lower its temperature, you will often trap the system in a particularly well organized or otherwise favorable state (e.g. crystallized, strengthened, altered conductivity). The process is known as annealing. An interesting use of MCMC is to seek minimums of a cost function ϕ by *simulating* annealing. Given ϕ, define an associated probability distribution

$$p_T(x) = \frac{1}{Z_T} e^{-\frac{1}{T}\phi(x)}$$

where $T \geq 0$ represents absolute temperature. For fixed values of T, the larger the likelihood of x the smaller the value of $\phi(x)$. We can seek local or even global minimums by slowly lowering T ($T = T(t) \downarrow 0$) with each iteration of a Gibbs sampler. For example, if ϕ has a single global minimum at x_o, and if the rate of cooling is sufficiently slow, then

$$\mathbb{P}\left(X(t) \to x_o\right) \to 1$$

as $t \to \infty$. Faster cooling leads to local (rather than global) minimums. Convergence is in probability but decidedly *not* almost sure. Indeed, the system will, provably, visit *every* state of x infinitely often!

iii. **(hidden states and Bayesian inference)** Suppose that we observe only some of the variables in a multivariate distribution: $p = p(x, y)$, where $x \in \mathbb{R}^d$, $y \in \mathbb{R}^{d'}$, and we only observe y. The rest of the variables, $x_1, x_2, \ldots, x_d$, are hidden. This comes up all the time. The observables might represent corrupted images, or tomographic images, and the goal is to restore the image or reconstruct the three-dimensional unobserved anatomy. Or perhaps only part of an image is visible and the problem is to *inpaint* the obscured portions of the image. Alternatively, the unobserved variables might be the parameters of a distribution, in which case we would think of $p(x) \doteq \sum_y p(x, y)$ as a *prior* distribution in a Bayesian formulation of the parameter estimation problem.

In these cases, it makes sense to seek samples from or expectations under the *posterior* distribution on x given the observed y: $p(x|y) = p(x, y)/p(y)$. In principle, there is nothing different about this problem from the ones we have already discussed. Since y is observed to have a particular value (say $y = \tilde{y}$), it is no longer random and can be treated as a constant. Our interest is in sampling (or computing expectations with respect to) $p(x|y) = p(x, \tilde{y})/p(\tilde{y})$, which is to say that we want to sample from the un-normalized distribution $p(x, \tilde{y})$. The normalization will typically be very hard to compute (if not intractable): $p(\tilde{y}) = \sum_x p(x, \tilde{y})$. But fortunately, normalization constants are almost always irrelevant to the evolution of $X(0), X(1), \ldots$ in an MCMC implementation, as we already saw in the case of sampling from the Ising model.

The point here is that sampling from $p(x|Y = \tilde{y})$ requires no special treatment. The algorithm is unchanged.

iv. **(the many flavors of MCMC)** The MCMC toolbox is large and sophisticated. Pedagogically, the Gibbs sampler is arguably the most straightforward, but different problems require different variations, and there are many to choose from. Gibbs sampling is a special case of an infinite collection of algorithms known as the Metropolis Hastings algorithms. Beyond that, there are newer approaches (e.g. Hamiltonian Monte Carlo), and others that rely on continuous, rather than discrete, sampling, such as running the Langevin equation or other types of stochastic differential equations.

DP in a Nutshell

To ease the notation, we will drop subscripts in writing probability mass functions, e.g. $p(x_A)$ is short for $p_{X_A}(x_A)$, $p(x_A|x_B)$ is short for $p_{X_A|X_B}(x_A|x_B)$, etc. Also, for a clique function, say $\phi_{\{u,v\}}(x_u, x_v)$, we will just write $\phi_{uv}(x_u, x_v)$.

It will be understood that $\prod_{c\in\tilde{\mathcal{C}}} \phi_c(x_c) = 1$ whenever $\tilde{\mathcal{C}} = \emptyset$ (i.e. when $\tilde{\mathcal{C}}$ contains no cliques).

The typical problem involves a set of observed variables $Y = Y_B$, and a set of latent (unobserved) variables $X = X_A$. Collectively, X and Y respect some graph $G(V, E)$, where $V = A \bigcup B$. We are interested in $p(x|y) = \mathbb{P}(X = x|Y = y)$, which respects the graph $G'(A, E')$, where G' is just the restriction of G to the subgraph defined on the vertices in A ("Conditioning Rule").

More generally we have a GRF distribution $p(x)$ wrt some graph G, which we will denote by $G = G(V, E)$, whether conditioned or not. Throughout, d will represent the dimensionality (number of components) of X: $d = |V|$.

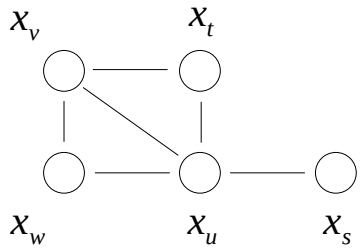
Computing Goals

- **a.** Compute most-likely (argmax) configuration, and its probability
- **b.** Compute marginals, expectations and normalizing constants
- **c.** Sampling
- **d.** Numerical stability

In all cases, the main idea is to exploit the conditional independencies captured by G. In what follows, we will carry along a specific graph structure as an example, using it repeatedly to illustrate the general formulas.

a. Most-likely (argmax) configuration, and its probability

e.g. $V = \{s, t, u, v, w\}$



$$p(x) = \phi_{uvw}(x_u, x_v, x_w)\phi_u(x_u)\phi_{ut}(x_u, x_t)\phi_{vt}(x_v, x_t)\phi_t(x_t)\phi_{us}(x_u, x_s)$$

(For now, we assume that the probability is normalized, possibly following the computations presented in section **b**, below.)

Steps to compute $\tilde{x} = \arg\max_x p(x)$

1. Absorb sub-cliques (optional, facilitates book keeping), and redefine $\mathcal{C}$ (set of relevant cliques) accordingly.

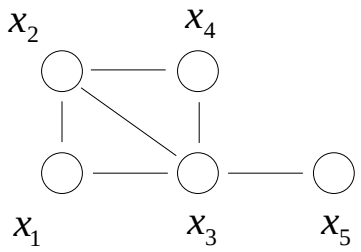
   e.g. $\psi_{uvw} = \phi_{uvw}$, $\psi_{ut} = \phi_u\phi_{ut}$, $\psi_{vt} = \phi_{vt}\phi_t$, and $\psi_{us} = \phi_{us}$

   $$\begin{aligned} p(x) &= \psi_{uvw}(x_u, x_v, x_w)\psi_{ut}(x_u, x_t)\psi_{vt}(x_v, x_t)\psi_{us}(x_u, x_s) \\ \mathcal{C} &= \{\{u, v, w\}, \{u, t\}, \{v, t\}, \{u, s\}\} \end{aligned}$$

   Remark Could do "better" with $\mathcal{C} = \{\{u, v, w\}, \{u, v, t\}, \{u, s\}\}$, but this hides some of the structure in the factorization.

2. Choose a site-visitation schedule (order of computation), and re-label sites.

   e.g.

   Now $V = \{1, 2, 3, 4, 5\}$ and

   $$\begin{aligned} p(x) &= \psi_{123}(x_1, x_2, x_3)\psi_{24}(x_2, x_4)\psi_{34}(x_3, x_4)\psi_{35}(x_3, x_5) \\ \mathcal{C} &= \{\{1, 2, 3\}, \{2, 4\}, \{3, 4\}, \{3, 5\}\} \end{aligned}$$

3. Define the boundary and the innovation cliques, for each $k = 1, 2, \ldots, d$.

   boundaries

   $$\begin{aligned} b_k &= \text{boundary of } 1, 2, \ldots, k \text{ in } G \\ &= \{l \in S : l > k \text{ and } l \in c \text{ for some } c \in \mathcal{C} \ni c \cap \{1, 2, \ldots, k\} \neq \emptyset\} \end{aligned}$$

   e.g. $b_1 = \{2, 3\}$, $b_2 = \{3, 4\}$, $b_3 = \{4, 5\}$, $b_4 = \{5\}$, $b_5 = \emptyset$

   innovation cliques

   $$\begin{aligned} \mathcal{C}_k &= \text{relevant new cliques at step } k \\ &= \{c \in \mathcal{C} : k \in c \text{ and } l \notin c \, \forall l < k\} \end{aligned}$$

   e.g. $\mathcal{C}_1 = \{\{1, 2, 3\}\}$, $\mathcal{C}_2 = \{\{2, 4\}\}$, $\mathcal{C}_3 = \{\{3, 4\}, \{3, 5\}\}$, $\mathcal{C}_4 = \emptyset$, $\mathcal{C}_5 = \emptyset$

4. Loop through sites, computing conditional optima.

   initialize:

$$\begin{aligned} R_1(x_{b_1}) &= \arg\max_{x_1} \prod_{c \in \mathcal{C}_1} \psi_c(x_c) \\ M_1(x_{b_1}) &= \max_{x_1} \prod_{c \in \mathcal{C}_1} \psi_c(x_c) \\ &= \prod_{c \in \mathcal{C}_1} \psi_c(x_c) \Big| \; x_1 = R_1(x_{b_1}) \end{aligned}$$

   e.g.

$$\begin{aligned} R_1(x_2, x_3) &= \arg\max_{x_1} \psi_{123}(x_1, x_2, x_3) \\ M_1(x_2, x_3) &= \psi_{123}(R_1(x_2, x_3), x_2, x_3) \end{aligned}$$

   iterate:

$$\begin{aligned} R_k(x_{b_k}) &= \arg\max_{x_k} M_{k-1}(x_{b_{k-1}}) \prod_{c \in \mathcal{C}_k} \psi_c(x_c) \\ M_k(x_{b_k}) &= \max_{x_k} M_{k-1}(x_{b_{k-1}}) \prod_{c \in \mathcal{C}_k} \psi_c(x_c) \\ &= M_{k-1}(x_{b_{k-1}}) \prod_{c \in \mathcal{C}_k} \psi_c(x_c) \Big| \; x_k = R_k(x_{b_k}) \end{aligned}$$

   $k = 2, 3, \ldots, d$. (Break ties arbitrarily.)

   e.g.

$$\begin{aligned} R_2(x_3, x_4) &= \arg\max_{x_2} M_1(x_2, x_3)\psi_{24}(x_2, x_4) \\ M_2(x_3, x_4) &= M_1(R_2(x_3, x_4), x_3)\psi_{24}(R_2(x_3, x_4), x_4) \\ R_3(x_4, x_5) &= \arg\max_{x_3} M_2(x_3, x_4)\psi_{34}(x_3, x_4)\psi_{35}(x_3, x_5) \\ M_3(x_4, x_5) &= M_2(R_3(x_4, x_5), x_4)\psi_{34}(R_3(x_4, x_5), x_4)\psi_{35}(R_3(x_4, x_5), x_5) \\ R_4(x_5) &= \arg\max_{x_4} M_3(x_4, x_5) \\ M_4(x_5) &= M_3(R_4(x_5), x_5) \\ R_5 &= \arg\max_{x_5} M_4(x_5) \\ M_5 &= M_4(R_5) \end{aligned}$$

5. Compute global optimum, "backwards."

$$\tilde{x}_d = R_d \quad \tilde{x}_{k-1} = R_{k-1}(\tilde{x}_{b_{k-1}}), \; k = d, ..., 2$$

   e.g. $\tilde{x}_5 = R_5$, $\tilde{x}_4 = R_4(\tilde{x}_5)$, $\tilde{x}_3 = R_3(\tilde{x}_4, \tilde{x}_5)$, $\tilde{x}_2 = R_2(\tilde{x}_3, \tilde{x}_4)$, $\tilde{x}_1 = R_1(\tilde{x}_2, \tilde{x}_3)$

Remarks

1. If p is not normalized, $p(x) = \frac{1}{Z}\prod_{c\in\mathcal{C}}\phi_C(x_c)$ and Z is unknown, then we can just ignore Z; $\tilde{x}$ is still the most-likely configuration.

2. In any case, if $p(x) = \frac{1}{Z}\prod_{c\in\mathcal{C}}\phi_C(x_c)$ then $p(\tilde{x}) = \frac{1}{Z}M_d$.

3. An easy modification gives the n best, $n > 1$.

4. Computational cost: Let $S = max_{i=1:d}|\Omega_i|$. Step k involves (at most) $O(S^{|b_k|+1})$ operations. (It could be less, if $\{1, 2, \ldots, k\}$ breaks into two or more disjoint components or if $|\Omega_k| < S$.) Hence the cost of the entire procedure is no more than

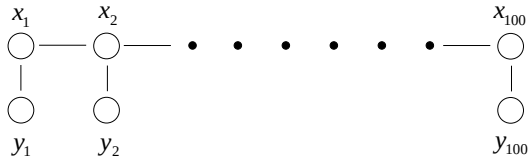
$$O(d \cdot S^{B+1})$$

   where $B = \max_k |b_k|$. (A similar bound works for memory, but with exponent B instead of $B + 1$.) The brute-force cost is $O(S^d)$.

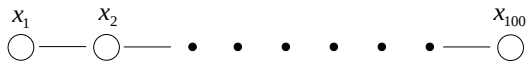
   e.g. $B = 2$ in the example. If the visitation schedule were re-ordered, exchanging 3 and 4, B would still be 2, but the third step would require $O(S^2)$ instead of $O(S^3)$ operations. In either case, cost is about $5S^3$. On the other hand, starting at x_u (i.e. what we've relabeled as x_3) would be a bad idea, costing S^5 operations right off the bat.

5. The ordering that minimizes B is NP hard.

6. Hidden Markov Model (HMM)



   Hence $p(x|y)$ is first-order Markov:



   Using the left-to-right visitation schedule, $B = 1$ and the number of computations is $O(100S^2)$. Compare to the brute-force maximization: $O(S^{100})$. With common state space $\{0, 1\}$ for example, DP costs about 400 computations instead of about 10^{30}.

7. Look at it this way:

$$\begin{aligned}\max_x p(x) &= \max_{x_5}\max_{x_4}\max_{x_3}\max_{x_2}\max_{x_1}\psi_{123}\psi_{24}\psi_{34}\psi_{35}\\ &= \max_{x_5}\max_{x_4}\max_{x_3}\psi_{34}\psi_{35}\max_{x_2}\psi_{24}\underbrace{\max_{x_1}\psi_{123}}_{M_1(x_2,x_3)}\\ &\qquad\qquad\qquad\qquad\qquad\qquad\underbrace{\qquad\qquad}_{M_2(x_3,x_4)}\\ &\qquad\qquad\qquad\qquad\underbrace{\qquad\qquad\qquad\qquad}_{M_3(x_4,x_5)}\\ &\qquad\qquad\qquad\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad}_{M_4(x_5)}\\ &\qquad\qquad\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{M_5}\end{aligned}$$

8. When d is large there can be (solvable) overflow and underflow issues. One solution that generally works is to replace all clique functions by their logarithms, and all products by their sums.

   e.g. For any clique $c \in \mathcal{C}$ write $\tilde{\psi}_c$ for $\log \psi_c$.

$$\begin{aligned}R_1(x_2,x_3) &= \arg\max_{x_1}\tilde{\psi}_{123}(x_1,x_2,x_3)\\ M_1(x_2,x_3) &= \tilde{\psi}_{123}(R_1(x_2,x_3),x_2,x_3)\\ R_2(x_3,x_4) &= \arg\max_{x_2}\big(M_1(x_2,x_3)+\tilde{\psi}_{24}(x_2,x_4)\big)\\ M_2(x_3,x_4) &= M_1(R_2(x_3,x_4),x_3)+\tilde{\psi}_{24}(R_2(x_3,x_4),x_4)\\ R_3(x_4,x_5) &= \arg\max_{x_3}\big(M_2(x_3,x_4)+\tilde{\psi}_{34}(x_3,x_4)+\tilde{\psi}_{35}(x_3,x_5)\big)\\ M_3(x_4,x_5) &= M_2(R_3(x_4,x_5),x_4)+\tilde{\psi}_{34}(R_3(x_4,x_5),x_4)+\tilde{\psi}_{35}(R_3(x_4,x_5),x_5)\\ R_4(x_5) &= \arg\max_{x_4}M_3(x_4,x_5)\\ M_4(x_5) &= M_3(R_4(x_5),x_5)\\ R_5 &= \arg\max_{x_5}M_4(x_5)\\ M_5 &= M_4(R_5)\end{aligned}$$

   $\tilde{x}_5 = R_5$, $\tilde{x}_4 = R_4(\tilde{x}_5)$, $\tilde{x}_3 = R_3(\tilde{x}_4,\tilde{x}_5)$, $\tilde{x}_2 = R_2(\tilde{x}_3,\tilde{x}_4)$, $\tilde{x}_1 = R_1(\tilde{x}_2,\tilde{x}_3)$

   And if p is normalized, then $p(\tilde{x}) = \exp(M_5)$.

b. Marginals, expectations, and normalizing constant

e.g. Same example, but not normalized:

$$\begin{aligned}p(x) &\propto \phi_{uvw}(x_u,x_v,x_w)\phi_u(x_u)\phi_{ut}(x_u,x_t)\phi_{vt}(x_v,x_t)\phi_t(x_t)\phi_{us}(x_u,x_s)\\ &= \frac{1}{Z}\phi_{uvw}(x_u,x_v,x_w)\phi_u(x_u)\phi_{ut}(x_u,x_t)\phi_{vt}(x_v,x_t)\phi_t(x_t)\phi_{us}(x_u,x_s)\end{aligned}$$

Compute

$$Z = \sum_x \phi_{uvw}(x_u,x_v,x_w)\phi_u(x_u)\phi_{ut}(x_u,x_t)\phi_{vt}(x_v,x_t)\phi_t(x_t)\phi_{us}(x_u,x_s)$$

Essentially the same as computing optimal configurations:

Steps to compute $\sum_x \prod_{c\in\mathcal{C}} \phi_c(x_c)$

1. Absorb sub-cliques (optional, facilitates book keeping), and redefine $\mathcal{C}$ (set of relevant cliques) accordingly (as in **a**, above).

   e.g. $\psi_{uvw} = \phi_{uvw}$, $\psi_{ut} = \phi_u\phi_{ut}$, $\psi_{vt} = \phi_{vt}\phi_t$, and $\psi_{us} = \phi_{us}$

   $$\begin{aligned} p(x) &= \frac{1}{Z}\psi_{uvw}(x_u, x_v, x_w)\psi_{ut}(x_u, x_t)\psi_{vt}(x_v, x_t)\psi_{us}(x_u, x_s) \\ \mathcal{C} &= \{\{u, v, w\}, \{u, t\}, \{v, t\}, \{u, s\}\} \end{aligned}$$

2. Choose a site-visitation schedule (order of computation), and re-label sites (as in **a**, above).
   e.g. $V = \{1, 2, 3, 4, 5\}$ (as in **a**) and

   $$\begin{aligned} p(x) &= \frac{1}{Z}\psi_{123}(x_1, x_2, x_3)\psi_{24}(x_2, x_4)\psi_{34}(x_3, x_4)\psi_{35}(x_3, x_5) \\ \mathcal{C} &= \{\{1, 2, 3\}, \{2, 4\}, \{3, 4\}, \{3, 5\}\} \end{aligned}$$

3. Define the boundary and the innovation cliques, for each $k = 1, 2, \ldots, d$ (as in **a**, above).

4. Loop through sites, computing partial sums.
   initialize:

   $$T_1(x_{b_1}) = \sum_{x_1} \prod_{c\in\mathcal{C}_1} \psi_c(x_c)$$

   e.g.

   $$T_1(x_2, x_3) = \sum_{x_1} \psi_{123}(x_1, x_2, x_3)$$

   iterate:

   $$T_k(x_{b_k}) = \sum_{x_k} T_{k-1}(x_{b_{k-1}}) \prod_{c\in\mathcal{C}_k} \psi_c(x_c)$$

   $k = 2, 3, \ldots, d$.
   Then $Z = T_d = \sum_x \prod_{c\in\mathcal{C}} \psi_c(x_c)$.
   e.g.

   $$\begin{aligned} T_2(x_3, x_4) &= \sum_{x_2} T_1(x_2, x_3)\psi_{24}(x_2, x_4) \\ T_3(x_4, x_5) &= \sum_{x_3} T_2(x_3, x_4)\psi_{34}(x_3, x_4)\psi_{35}(x_3, x_5) \\ T_4(x_5) &= \sum_{x_4} T_3(x_4, x_5) \\ T_5 &= \sum_{x_5} T_4(x_5) \\ &= Z \end{aligned}$$

Remarks

1. Look at it this way:

$$
\begin{aligned}
Z &= \sum_{x_5}\sum_{x_4}\sum_{x_3}\sum_{x_2}\sum_{x_1}\psi_{123}\psi_{24}\psi_{34}\psi_{35} \\
&= \sum_{x_5}\sum_{x_4}\sum_{x_3}\psi_{34}\psi_{35}\sum_{x_2}\psi_{24}\underbrace{\sum_{x_1}\psi_{123}}_{T_1(x_2,x_3)} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\underbrace{\qquad\qquad\qquad\qquad}_{T_2(x_3,x_4)} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\underbrace{\qquad\qquad\qquad\qquad\qquad}_{T_3(x_4,x_5)} \\
&\qquad\qquad\qquad\qquad\qquad\qquad\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad}_{T_4(x_5)} \\
&\qquad\qquad\qquad\qquad\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{T_5}
\end{aligned}
$$

2. (expectation) If $p(x) = \prod_{c\in\mathcal{C}}\psi_c(x_c)$ and $J(x) = J(x_A)$, for some function J and sites $A \subseteq V$, then $E_p[J(X_A)] = \sum_x J(x_A)\prod_{c\in\mathcal{C}}\psi_c(x_c)$, which is the same as the normalization problem, but with the set of cliques, $\mathcal{C}$, augmented with the clique A.

3. Computational complexity: $k'th$ step costs $O(S^{b_k+1})$ and the entire problem costs $O(dS^{B+1})$, $B = \max_k |b_k|$, the same as computing the most-likely configuration.

4. (marginals) Let $p(x) = \frac{1}{Z}\prod_{c\in\mathcal{C}}\psi_c(x_c)$. Notice (by induction) that

$$
T_k(x_{b_k}) = \sum_{x_{1:k}} \prod_{\substack{c\in\mathcal{C} \\ c\,\cap\,\{1,2,\ldots,k\}\neq\emptyset}} \psi_c(x_c), \quad k = 1,2,\ldots,d
$$

   Hence, for $k = 1,2,\ldots,d-1$

$$
\begin{aligned}
p(x_{k+1:d}) &= \sum_{x_{1:k}} p(x) \\
&= \frac{1}{Z}\sum_{x_{1:k}}\prod_{c\in\mathcal{C}}\psi_c(x_c) \\
&= \frac{1}{Z}\Big(\prod_{\substack{c\in\mathcal{C} \\ c\,\cap\,\{1,2,\ldots,k\}=\emptyset}} \psi_c(x_c)\Big)\sum_{x_{1:k}}\prod_{\substack{c\in\mathcal{C} \\ c\,\cap\,\{1,2,\ldots,k\}\neq\emptyset}} \psi_c(x_c) \\
&= \frac{1}{Z}\Big(\prod_{\substack{c\in\mathcal{C} \\ c\,\cap\,\{1,2,\ldots,k\}=\emptyset}} \psi_c(x_c)\Big)T_k(x_{b_k})
\end{aligned}
$$

   (As for $p(x_{1:d})$: $p(x_{1:d}) = p(x) = \frac{1}{Z}\prod_{c\in\mathcal{C}}\psi_c(x_c)$.)

e.g.

$$\begin{aligned}
p(x_1,x_2,x_3,x_4,x_5) &= \frac{1}{Z}\psi_{123}(x_1,x_2,x_3)\psi_{24}(x_2,x_4)\psi_{34}(x_3,x_4)\psi_{35}(x_3,x_5)\\
p(x_2,x_3,x_4,x_5) &= \frac{1}{Z}\psi_{24}(x_2,x_4)\psi_{34}(x_3,x_4)\psi_{35}(x_3,x_5)T_1(x_2,x_3)\\
p(x_3,x_4,x_5) &= \frac{1}{Z}\psi_{34}(x_3,x_4)\psi_{35}(x_3,x_5)T_2(x_3,x_4)\\
p(x_4,x_5) &= \frac{1}{Z}T_3(x_4,x_5)\\
p(x_5) &= \frac{1}{Z}T_4(x_5)
\end{aligned}$$

5. There are tools for computing marginal distributions of arbitrary subsets of the variables. These use various methods to organize the computations so as to avoid repeating sub-computations (e.g. see "triangulation," "junction trees," etc.).

c. Sampling

If $p(x) = \frac{1}{Z}\prod_{c\in\mathcal{C}}\psi_c(x_c)$, then by remark 4 above, for $k = 2, 3, \ldots, d-1$

$$\begin{aligned}
p(x_k|x_{k+1:d}) &= \frac{p(x_{k:d})}{p(x_{k+1:d})}\\
&= \frac{(\prod_{\substack{c\in\mathcal{C}\\ c\,\cap\,\{1,2,\ldots,k-1\}=\emptyset}}\psi_c(x_c))T_{k-1}(x_{b_{k-1}})}{(\prod_{\substack{c\in\mathcal{C}\\ c\,\cap\,\{1,2,\ldots,k\}=\emptyset}}\psi_c(x_c))T_k(x_{b_k})}\\
&= \frac{(\prod_{c\in\mathcal{C}_k}\psi_c(x_c))T_{k-1}(x_{b_{k-1}})}{T_k(x_{b_k})}
\end{aligned}$$

and, when $k = 1$,

$$p(x_1|x_{2:d}) = \frac{\prod_{c\in\mathcal{C}}\psi_c(x_c)}{(\prod_{\substack{c\in\mathcal{C}\\ c\,\cap\,\{1\}=\emptyset}}\psi_c(x_c))T_1(x_{b_1})} = \frac{\prod_{c\in\mathcal{C}_1}\psi_c(x_c)}{T_1(x_{b_1})}$$

Also,

$$\begin{aligned}
p(x_d) &= \frac{1}{Z}(\prod_{c\in\mathcal{C}_d}\psi_c(x_c))T_{d-1}(x_{b_{d-1}})\\
&= \frac{1}{Z}(\prod_{c\in\mathcal{C}_d}\psi_c(x_c))T_{d-1}(x_d)
\end{aligned}$$

Finally, since

$$p(x) = p(x_d)p(x_{d-1}|x_d)p(x_{d-2}|x_{d-1:d})p(x_{d-3}|x_{d-2:d})\cdots p(x_1|x_{2:d})$$

and since we have explicit representations for $p(x_d)$ and $p(x_k|x_{k+1:d})$ $\forall k = 1, 2, \ldots, d-1$, we can sample from p:

$$\begin{aligned}
\tilde{x}_d &\sim p(x_d)\\
\tilde{x}_k &\sim p(x_k|\tilde{x}_{k+1:d}) \quad k = d-1, d-2, \ldots, 1
\end{aligned}$$

Remark Observe that $p(x_k|x_{k+1:d})$ depends only on x_k and x_{b_k}. Hence $p(x_k|x_{k+1:d}) = p(x_k|x_{b_k})$.
e.g.

$$\begin{aligned}
p(x_5) &= \frac{1}{Z}T_4(x_5) \\
p(x_4|x_5) &= \frac{T_3(x_4,x_5)}{T_4(x_5)} \\
p(x_3|x_4,x_5) &= \frac{\psi_{34}(x_3,x_4)\psi_{35}(x_3,x_5)T_2(x_3,x_4)}{T_3(x_4,x_5)} \\
p(x_2|x_3,x_4,x_5) &= \frac{\psi_{24}(x_2,x_4)T_1(x_2,x_3)}{T_2(x_3,x_4)} \\
&= p(x_2|x_3,x_4) \text{ (since there is no dependency on } x_5) \\
p(x_1|x_2,x_3,x_4,x_5) &= \frac{\psi_{123}(x_1,x_2,x_3)}{T_1(x_2,x_3)} \\
&= p(x_1|x_2,x_3) \text{ (since there is no dependency on } x_4 \text{ or } x_5)
\end{aligned}$$

And to sample:

$$\begin{aligned}
\tilde{x}_5 &\sim p(x_5) \\
\tilde{x}_4 &\sim p(x_4|\tilde{x}_5) \\
\tilde{x}_3 &\sim p(x_3|\tilde{x}_4,\tilde{x}_5) \\
\tilde{x}_2 &\sim p(x_2|\tilde{x}_3,\tilde{x}_4) \\
\tilde{x}_1 &\sim p(x_1|\tilde{x}_2,\tilde{x}_3)
\end{aligned}$$

d. Numerical stability

We have already discussed the advisability of using logarithms and addition in place of clique functions and products, in computing the most-likely configurations when d is large—see remark 8, in §**a**.

When d is large there can also be problems computing the functions $T_1, \ldots, T_d$, which are central to computing Z as well as to computing the conditional probabilities $p(x_k|x_{k+1:d})$ used in sampling. These functions are summations of products of many terms, and overflow or underflow (i.e. exceeding the capacity of the bit-based representation of numbers in your computer) can occur. In this final section we will revisit the computation of Z and the approach to sampling, using a numerical trick to recursively re-normalize $T_1, \ldots, T_d$ so as to avoid values that are too large or too small to be represented accurately.

Referring to the discussion in §**b**, and to step 4 in particular, consider replacing the recursion for $T_1, \ldots, T_d$

by the followng:

initialize: define $b_0 = \emptyset$ and $\tilde{T}_0(x_{b_0}) = 1$

iterate: for $k = 1, 2, \ldots, d$

$$\begin{aligned}
Temp(x_{b_k}) &= \sum_{x_k} \tilde{T}_{k-1}(x_{b_{k-1}}) \prod_{c \in \mathcal{C}_k} \psi_c(x_c) \\
S_k &= \sum_{x_{b_k}} Temp(x_{b_k}) \qquad \text{(if } b_k = \emptyset \text{ then } S_k = Temp\text{)} \\
\tilde{T}_k(x_{b_k}) &= Temp(x_{b_k})/S_k \qquad \text{(if } b_k = \emptyset \text{ then } \tilde{T}_k = 1\text{)}
\end{aligned}$$

e.g.

$$\begin{aligned}
Temp(x_2, x_3) &= \sum_{x_1} \psi_{123}(x_1, x_2, x_3) \\
S_1 &= \sum_{x_2, x_3} Temp(x_2, x_3) \\
\tilde{T}_1(x_2, x_3) &= Temp(x_2, x_3)/S_1 \\
Temp(x_3, x_4) &= \sum_{x_2} \tilde{T}_1(x_2, x_3)\psi_{24}(x_2, x_4) \\
S_2 &= \sum_{x_3, x_4} Temp(x_3, x_4) \\
\tilde{T}_2(x_3, x_4) &= Temp(x_3, x_4)/S_2 \\
Temp(x_4, x_5) &= \sum_{x_3} \tilde{T}_2(x_3, x_4)\psi_{34}(x_3, x_4)\psi_{35}(x_3, x_5) \\
S_3 &= \sum_{x_4, x_5} Temp(x_4, x_5) \\
\tilde{T}_3(x_4, x_5) &= Temp(x_4, x_5)/S_3 \\
Temp(x_5) &= \sum_{x_4} \tilde{T}_3(x_4, x_5) \\
S_4 &= \sum_{x_5} Temp(x_5) \\
\tilde{T}_4(x_5) &= Temp(x_5)/S_4 \\
Temp &= \sum_{x_5} \tilde{T}_4(x_5) \\
S_5 &= Temp \\
\tilde{T}_5 &= 1
\end{aligned}$$

Remarks

1. Every value of $\tilde{T}_k(x_{b_k})$ is between zero and one, and $\tilde{T}_d = 1$.

2. Using induction, it is not hard to show that

$$\tilde{T}_k(x_{b_k}) = \frac{\sum_{x_{1:k}} \prod_{\substack{c \in \mathcal{C} \\ c \cap \{1,2,\ldots,k\} \neq \emptyset}} \psi_c(x_c)}{\sum_{x_{b_k}} \sum_{x_{1:k}} \prod_{\substack{c \in \mathcal{C} \\ c \cap \{1,2,\ldots,k\} \neq \emptyset}} \psi_c(x_c)}$$

   (including when $k = d$, since $b_d = \emptyset$ and $\tilde{T}_d = 1$)

3. To recover the functions $T_1, T_2, \ldots, T_d$ we note (again by induction) that

$$T_k(x_{b_k}) = \tilde{T}_k(x_{b_k}) \prod_{j=1}^{k} S_j$$

4. (normalizing) Since $Z = \sum_x \prod_{c \in \mathcal{C}} \psi_c(x_c) = T_d$, and since $\tilde{T}_d = 1$, we get $Z = \prod_{j=1}^{d} S_j$. Z might be very large or very small, but $\log Z = \sum_{j=1}^{d} \log S_j$ will generally not cause any numerical problems.

5. (sampling) Using the expressions derived in §**c**, above:

$$\begin{aligned} p(x_d) &= \frac{1}{Z}\Big(\prod_{c \in \mathcal{C}_d} \psi_c(x_c)\Big) T_{d-1}(x_d) \\ &= \frac{\big(\prod_{c \in \mathcal{C}_d} \psi_c(x_c)\big) \tilde{T}_{d-1}(x_d) \prod_{j=1}^{d-1} S_j}{\prod_{j=1}^{d} S_j} \\ &= \frac{\big(\prod_{c \in \mathcal{C}_d} \psi_c(x_c)\big) \tilde{T}_{d-1}(x_d)}{S_d} \end{aligned}$$

   For $k = 2, \ldots, d-1$:

$$\begin{aligned} p(x_k | x_{k+1:d}) &= \frac{\big(\prod_{c \in \mathcal{C}_k} \psi_c(x_c)\big) T_{k-1}(x_{b_{k-1}})}{T_k(x_{b_k})} \\ &= \frac{\big(\prod_{c \in \mathcal{C}_k} \psi_c(x_c)\big) \tilde{T}_{k-1}(x_{b_{k-1}}) \prod_{j=1}^{k-1} S_j}{\tilde{T}_k(x_{b_k}) \prod_{j=1}^{k} S_j} \\ &= \frac{\big(\prod_{c \in \mathcal{C}_k} \psi_c(x_c)\big) \tilde{T}_{k-1}(x_{b_{k-1}})}{\tilde{T}_k(x_{b_k}) S_k} \end{aligned}$$

   And, finally:

$$p(x_1 | x_{2:d}) = \frac{\prod_{c \in \mathcal{C}_1} \psi_c(x_c)}{T_1(x_{b_1})} = \frac{\prod_{c \in \mathcal{C}_1} \psi_c(x_c)}{\tilde{T}_1(x_{b_1}) S_1}$$

e.g.

$$
\begin{aligned}
p(x_5) &= \frac{\tilde{T}_4(x_5)}{S_5} \\
p(x_4|x_5) &= \frac{\tilde{T}_3(x_4,x_5)}{\tilde{T}_4(x_5)S_4} \\
p(x_3|x_4,x_5) &= \frac{\psi_{34}(x_3,x_4)\psi_{35}(x_3,x_5)\tilde{T}_2(x_3,x_4)}{\tilde{T}_3(x_4,x_5)S_3} \\
p(x_2|x_3,x_4) &= \frac{\psi_{24}(x_2,x_4)\tilde{T}_1(x_2,x_3)}{\tilde{T}_2(x_3,x_4)S_2} \\
p(x_1|x_2,x_3) &= \frac{\psi_{123}(x_1,x_2,x_3)}{\tilde{T}_1(x_2,x_3)S_1}
\end{aligned}
$$

And to sample:

$$
\begin{aligned}
\tilde{x}_5 &\sim p(x_5) \\
\tilde{x}_4 &\sim p(x_4|\tilde{x}_5) \\
\tilde{x}_3 &\sim p(x_3|\tilde{x}_4,\tilde{x}_5) \\
\tilde{x}_2 &\sim p(x_2|\tilde{x}_3,\tilde{x}_4) \\
\tilde{x}_1 &\sim p(x_1|\tilde{x}_2,\tilde{x}_3)
\end{aligned}
$$