Probabilistic models survey

vishvAs vAsuki

April 17, 2012

Based on [1].

Contents

Part I

Introduction

1 Modeling

1.1 Model/ hypothesize for predictive ability

Many natural phenomena are not deterministic. We often want concise explanations for observed phenomena. This will grant us great predictive ability. Probability theory aims to model and understand uncertainties such phenomena.

1.1.1 Realism vs conciseness

The more the model reflects reality, the better it is. Yet, as we want to avoid overfitting to limited data, we seek conciseness.
So, the art here is to define more and more general/ concise models, which also manage to model reality closely, whose parameters turn out to be easy to learn in practice.
For details about the modeling process and its use, see the statistical inference survey.

1.1.2 Frequentist vs Subjective interpretations

There are two ways of modeling uncertainty (or conversely, 'interpreting' probabilities).
Physical/ empirical/ frequentist probability reflects frequency with which it occurs, which is itself unknown in advance. In its view, probability is embedded in the universe - as exemplified by Quantum physics.
Subjective/ evidential/ epistemological/ Bayesian probability instead reflects uncertainty in an entity's estimation of the frequentist probability - so it is in a sense more honest.
These models of uncertainty lead to different but overlapping approaches to statistical inference - see statistics survey for details. But, in either case, the mathematical theory/ axiomatization is identical.

1.1.3 Sampling mechanism

Often the probability measure is fully defined using the sampling mechanism (modeled to suit the application domain). A few probability puzzles and paradoxes rely on ambiguity in the description of the sampling mechanism.
Counting is also often fundamental to the definition of probability mechanisms. That is considered in the probability theory survey.

1.1.3.1 Possible errors

When analyzing sampling mechanisms and probability errors, one should be rigorous in order to avoid errors - particularly when dealing with conditional probabilities. This is illustrated with examples elsewhere.

1.2 Using the models

For ideas about actually doing feature extraction (maybe using kernels), using these model families, selecting the right model(s) based on training data while avoiding overfitting, see statistics/ pattern recognition survey.

1.2.1 Effectiveness of simple models

It is a very good idea to start with the simplest models - despite the lack of sophistication, they yield surprisingly good results. Eg: Naive bayes model in classification, sequence independence assumption in sequential labeling tasks (eg: Part of speech tagging.)

2 Non probabilistic models

For non probabilistic and deterministic models of phenomena (Eg: SVD, LSI for relationship between terms and documents), see elsewhere.

Part II

Simple Random variable densities

Random variables and the transformations between them are important in modeling randomness.

3 Distribution of values

3.1 Specification and classes

A distribution is often specified by a pdf or a cdf involving certain parameters. Or it may be specified by a stochastic process generating some values: ie in terms of other other distributions.
Sometimes, the density specified need not even be proper (sum/ integrate to 1) to be useful: Eg: In applying the conditional probability inversion trick.

3.1.1 Notation

If the pdf of X is a member (identified by the parameter p_1) of the function family $\{f(p)\}$, we write $X \sim f(p_1)$.

3.1.2 Parameter types

Location parameters specify important points in the distribution: Eg: μ in $N(\mu, \sigma^2)$ distribution.
Scale parameters specify how spread-out the distribution is. A parameter s is a scale parameter if, having set the location parameter to 0, $CDF(x; ks) = CDF(x/k; s)$ Eg: h in $C(x; \mu, h)$ distribution, and σ in $N(\mu, \sigma^2)$.
All other parameters are called shape parameters.

3.1.3 Specify continuous distribution over bounded support

Take Indicator fn $I_{(a,b)}$: See algebra ref. So, if U(a,b): $f(x) = (b-a)^{-1} I_{(a,b)}(x)$. Not differentiable in boundaries.

3.2 Inference, Sampling from distribution

See randomized algorithms ref.

4 Discrete probability distributions

4.1 Coin toss distribution

$X \sim$ Bernoulli(p) when $range(X) = \{0, 1\}$ and $Pr(X = 1) = p$.
When $p = 1/2$, X is called a Rademacher RV.

4.1.1 Properties

$E[X] = p$. $Var[X] = p - p^2$. Same as Bin(1, p).

4.1.2 Odds of success

$\frac{p}{1-p}$. This function is used in gambling (see economics survey).

4.2 Multiple coin-toss

Aka Binomial distribution.
This is the probability of x successes in n trials.
$X \sim Bin(n, p)$ when $Pr(X = x) = \binom{n}{x} p^x (1-p)^{n-x}$, $range[X] = [0, ..n]$.

4.2.1 Properties

$X = \sum X_i$, where X_i is Bernoulli RV.
So, $E[X] = np$. $Var[X] = nVar[X_i] = np(1-p)$, even if X_i are only pairwise independent.
The plot of the pdf looks like a discrete version of the bell curve truncated at 0 and n.

4.2.2 Approximations

As $n \to \infty$, it may be approximated by Poisson distribution if $p \to 0$. It is approximated by Normal distribution if p fixed.

4.2.2.1 With exponential decay for sq deviation distribution

From central limit theorem, $\frac{X-np}{\sqrt{np(1-p)}}$ approaches N(0, 1) as $n \to \infty$. Good if $n > (1-p)\frac{\max p,(1-p)}{\min p,(1-p)}$.

4.2.3 Poisson distribution

$\lim_{n\to\infty} Bin(n, f(n)) = Poi(nf(n))$:
$\lim_{n\to\infty} Pr(X = x) = \lim_{n\to\infty, p\to 0} \frac{P(n,x)p^x(1-p)^{n-x}}{x!} = (np)^x e^{-np}/x!$. Number of events which occur in a given time interval: Models Rare events; arrival rates.
$E[Poi(\mu)] = \sum(\mu)^x e^{-\mu}/x! = \mu e^{-\mu} e^{\mu} = \mu$. Poi(m) + Poi(n) = Poi(m+n).
For Poi(m), $M_X(t) = e^{\mu(e^t-1)}$.
So, Chernoff bounds for Poi(m): Select $t = \ln(x/m)$ either +ve or -ve.
Looks like a discrete version of the bell curve, truncated at 0.

4.2.4 Random walk on a line

Consider a random walk of n steps on the number line with start position: 0. The position changes by +1 or -1 with each step.
In expectation, due to symmetry, the final position is 0. We want to find the expected deviation from 0 in the end.
To model the number of +1 steps taken, one can use $X \sim Bin(n, 0.5)$. $E[X] = n/2$, and $Var[X] = \sigma^2 = n/4$. $Pr(X \geq n/2(1 + \frac{k}{\sqrt{n}})) \leq 2e^{\frac{-k^2}{6}}$. So, whp, ye're within $O(\sqrt{n})$ of 0.

4.2.5 (balls, bins)

4.2.5.1 Process

Suppose that you threw n balls into m bins so that each ball fell into a random bin.

4.2.5.2 Distribution

Balls in some bin, $X_i \sim Bin(m, 1/n) \approx Poi(m/n)$.
$Pr(actual) = Pr(appr|\sum X_i = m)$; $Pr(actual) < e\sqrt{m}Pr(appr)$.
$Pr(X_i = 0) = (1 - 1/n)^m \approx e^{m/n}$; So E[num empty bins]= $ne^{m/n}$; No empty bins whp: $m = n \ln n + cn$ (Coupon collector). Also, $min \approx max$ whp: $m = n \ln n + cn$. [**Find proof**]
Max load when n=m: whp $\Theta(\ln n / \ln \ln n)$: upper bound by Chernoff; [**Incomplete**].
Birthday paradox : some bin has 2 balls: whp when $m = \sqrt{n}$: Pr (every ball in diff bin) = P = $\prod_{i=1}^{m}(1 - \frac{(i-1)}{n}) = P(n, m)/n^m \approx \prod_{i=1}^{m} e^{\frac{-(i+1)}{n}} \approx e^{-m^2/(2n)}$.
Power of $d \geq 2$ choices: $\Theta(\ln \ln n / \ln d)$. [**Find proof**]
(balls, bins) Strategies: Find probabilities of basic events; Combine; Approximate. Use poisson approximation.

4.3 Categorical distribution

Consider a trial with discrete and finite outcomes. Outcome i has probability p_i. The outcome can be modeled as a random variable or as a 1 of k random vector X. If latter, can write: $Pr(X = x) = \prod p_i^{x_i}$.

4.4 Multinomial distribution

Consider n trials - each with k (discrete and finite) outcomes. Consider the k-dimensional random vector X where X_i represents the number of times outcome i appears. Note that this implies $\sum_i X_i = n$.
This can be viewed as the distribution of the sum of sequence of n random vectors with categorical distribution. $Pr(x) = \binom{n}{x_1..x_k} \prod p_i^{x_i}$.

4.5 Geometric distribution

Probability of x successful trials before first failure. $X \sim Geom(p) : Pr(X = x) = (1 - p)^{x-1}p$. Geom(p) is memoryless. Gambler's fallacy. $E[geom(p)] = 1/p$. $Var[Geom(p)] = (1 - p)/p^2$.

4.6 Hypergeometric distribution

Parameters: N: number of balls in an urn. m: number of +ve balls. n: number of trials. You draw n balls, but, unlike binomial distribution, no replacement is allowed.
Probability of t successes: $Pr(X = k) = \frac{\binom{m}{k}\binom{N-m}{n-k}}{\binom{N}{n}}$.

4.7 Smoothing

4.7.1 Motivation

4.7.1.1 Incomplete knowledge of range

Suppose that you have a discrete valued random variable X, and that $ran(X)$ is not completely specified beforehand: perhaps due to limited sample S from f_X where only values $T \subseteq ran(X)$ were observed. Suppose we want to model the distribution f_X.
A basic model would be to use the empirical distribution $\hat{f}_X$. But, this model will assign probability 0 to any $x \in T$. So, we may want a model which does not do this.

4.7.1.2 Assumption of continuous ran(X)

In this case, $ran(X)$ is actually assumed to be part of a vector space, so that no finite sample can reveal the entire $ran(X)$. Examples of smoothing to deal with this case is described in the statistics survey (eg: kernel density estimation techniques).

4.7.2 Add 1

This tries to address the problem of 'limited observation of $ran(X)$'. Suppose that $\hat{f}_X(x) = \frac{S(x)}{|S|}$, where $S(x)$ represents the number of occurrences of x in the sample S with the observed range T, is the basic unsmoothed empirical probability model.
This can be smoothed by first adding an element a placeholder unk for unobserved values to T to get T', and then setting the distribution $f'_X(x) = \frac{S(x)+1}{|S|+|T'|} \forall x \in T$; note that $S(unk) = 0$. This ensures that $f'_X(x)$ sums to 1 and that $f'_X(unk) = \frac{1}{|S|+|T'|}$.

4.7.2.1 Add k

This generalizes add 1 smoothing by doing this: $f'_X(x) = \frac{S(x)+k}{|S|+k|T'|}$.
$f'_X(unk)$ increases with k, as $\frac{l+k}{m+k} - \frac{l}{m} > 0$.

4.7.3 Different additions

One generalization of add-k smoothing could be to let: $f'_X(x) = \frac{S(x)+k(x)}{|S|+\sum_x k(x)}$.
Then, $f'_X(unk)$, if included, would be $\frac{k(unk)}{|S|+\sum_x k(x)}$.
Thus, this tries to compensate for limitations in sample size using some alternative model (represented by k(x)) and can accommodate an unobserved event.

4.7.3.1 Use backoff probabilities

This can be viewed from the perspective of backoff probabilities. Suppose there is an estimate g_X different from $f_X(x) = \frac{S(x)}{|S|}$, and suppose that $\sum_x k(x) = K$. We can model $f'_X = \frac{S(x)+Kg_X(x)}{|S|+K}$.

5 Mode-deviation penalizers

For the important exponential decay for squared deviation from mean, see exponential distribution family chapter.
Exponential decay and bilateral-exponential decay distributions are described elsewhere.

5.1 Inverse squared decay

Aka Cauchy distribution. Parameters: mean/ mode μ, height-at-mean parameter h. The pdf is $C(x;\mu,h) = \pi^{-1}(\frac{h}{(x-\mu)^2+h^2})$.

5.1.1 Limited to positive deviation

Aka half-Cauchy distribution. Parameters: mean/ mode m, height-at-mean parameter h. Range of the random variable is $[m,\infty]$. The pdf is $C(x;\mu,h) = 2\pi^{-1}(\frac{h}{(x-m)^2+h^2})$.

6 Exponential families

6.1 Exponential family of distributions

6.1.1 Generated by h and feature function, parametrized by t

$x \in X \subseteq R^d$; Base measure $h : R^d \to R$, feature extraction fn $\phi : R^d \to R^k$. $p(x;t) \propto h(x)e^{\langle t,\phi(x)\rangle}$. Exponential family: $\{p(x;t)\forall t\}$. $\phi_i()$ aka potential functions.
Can be written as $p(x;t) = h(x)e^{\langle t,\phi(x)\rangle - G(t)} = Z^{-1}h(x)e^{\langle t,\phi(x)\rangle}$, where $G(t) = \log\sum_X h(x)e^{\langle t,\phi(x)\rangle}$ is the log partition fn.
$G(t)$ mayn't always exist for any t [**Find proof**], so define natural parameter space: $N = \left\{t \in R^k : -1 < G(t) < 1\right\}$.

6.1.1.1 Canonical form.

Pick natural parameters such that j(t)=kt for any k: so non unique. Natural parameter space is convex. [**Find proof**]

6.1.1.2 Minimal parametrization.

If the features $\phi_i(x)$ are not linearly independent, the exponential family is overparametrized. So, the same probability distribution can be expressed using multiple distinct parameter-vectors.

6.1.2 Undirected graphical model from exp family distribution

Let $h(x) = 1$. $e^{\langle t,\phi(x)\rangle} = \prod_i e^{t_i\phi_i(x)}$: make cliques among terms involved in $\phi_i(x)$.

6.1.3 Maximum entropy distribution with given means

6.1.3.1 The optimization problem

$\max_p H(x) : E_{x\sim p}[\phi(x)] = \mu, p \geq 0, 1^T p = 1$.
Equivalently, can use $H(x) = |dom(X)| - KL(p, U)$ in the objective: so finding p closest to U with the given means.

6.1.3.2 Lagrangian form

$\max_p \sum_x p(x) log(\frac{1}{p(x)}) + \sum_i t_i E_{x\sim p}[\phi_i(x)] = \mu_i, p \geq 0, 1^T p = 1$; reduce it to $\max -KL(p, p^*) + \log Z$, where $p^*(x) = Z^{-1} e^{-\sum \sum_i t_i(\phi_i(x)-\mu_i)}$.
So, the solution is a member of the exponential family generated by the base measure U and feature functions $\phi()$.

6.1.3.3 Closest distribution to h with given means

Solve $\min_p KL(h, p) : E_{x\sim p}[\phi(x)] = \mu, p \geq 0, 1^T p = 1$ similarly to show that solution belongs to exponential family generated by h and ϕ.

6.1.3.4 Parametrization by means

t then corresponds to the lagrange multipliers; whose value depends on μ; So, a distribution in the family can equivalently be parametrized by means μ.

Polytope of means The set of all possible means forms a polytope; and finding a distribution from an exponential family G then often viewed as finding a point μ in this polytope: see statistics ref.
Any μ corresponds to some p.

6.2 Inverse Exponential decay for squared deviation from mean

Aka Normal distribution.

6.2.1 Importance

The 'bell curve' is often observed in nature. 'Normal distribution' happens to be a suitable small tailed distribution model for these phenomena.
It is also important due to the Central Limit Theorem: the estimator of the mean approaches the normal distribution as the number of samples increases.

6.2.2 1D case

6.2.2.1 pdf, cdf

$X \sim N[\mu, \sigma^2]$: a location parameter and a scale parameter. Range(X) = R. Probability density (Gaussian)
$N[x|\mu, \sigma^2] = f_{\mu,\sigma^2}(X = x) = \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{(x-\mu)^2}{2\sigma^2})$.
Defined this way to ensure symmetry about μ, the mean: The bell curve; inverse exponential decay away from the mean;
$\frac{1}{2\sigma\pi}$ factor to ensure that $\int_{-\infty}^{\infty} N[x|\mu, \sigma^2]dx = 1$ (aka Normalization), using Gaussian integral.
Thence confirm using direct integration: $\int N[x|\mu, \sigma^2]xdx = \mu$. Also, using integration by parts, $var[X] = E[X^2] - E[X]^2 = \sigma^2$. Mode coincides with the mean.

Important densities $Pr(|X - \mu| \leq \sigma) \approx .68$.
$Pr(|X - \mu| \leq 2\sigma) \approx .95$.
$Pr(|X - \mu| \leq 3\sigma) \approx .997$.

6.2.2.2 Standard Normal distribution

$N(x|0, 1) = \frac{1}{2\pi} exp(-\frac{(x)^2}{2})$. Aka Z distribution. Very convenient as every normal distribution can be viewed as a standard normal rescaled and shifted.

6.2.2.3 CDF calculation

CDF $F(x) = \int_{-\infty}^{x} f(x)dx$ looks like sigmoid fn curve, but has no closed form. So, to calculate CDF, convert to standard normal distribution, look up corresponding entry in table. So, Inverse Exponential tail bounds hold. [**Check**]
Often convert $X \sim N(\mu, \sigma^2) \rightarrow (\frac{X-\mu}{\sigma}) \sim N(0, 1)$ to use N(0, 1) CDF table to find CDF of X.
In Matlab, can use erfc and erfcinv. $F(x) = 1/2erfc(-u/\sqrt{2})$.

6.2.2.4 Moment generating function

$M(t) = E[e^{tX}] = \int e^{tx} \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{(x-\mu)^2}{2\sigma^2})dx$
$= \int e^{t\mu+\sigma^2t^2/2} \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{(x-\mu-\sigma^2t)^2}{2\sigma^2})dx$
$= e^{t\mu+\sigma^2t^2/2}$ by completing the squares.

6.2.2.5 Other properties

If $\{X_i\}$ are iid $N(\mu, \sigma^2)$,
using mgf, $\sum a_i X_i \sim N(\sum a_i \mu, \sum a_i^2 \sigma^2)$.
Log concave distribution is close to Normal: Just visualize.
If $X \sim N[0,1]$, $f_{0,1}$ is eigenfunction of the Fourier transform.
Also, $E[e^{sX^2}] = (1-2s)^{0.5}$ [**Find proof**].

6.2.3 Multidimensional case

6.2.3.1 Definition with univariates

$X \in R^n$ has multidimensional normal distribution if $\forall a \in R^n : a^T X$ has a univariate normal distribution.
It is determined compeltely by μ and covariance matrix Σ, as for any random vector X, $a^T X$ has mean $a^T \mu$ and variance $a^T \Sigma a$.
So, any subvector Y also has multivariate normal. ***So, all marginals are normal!***

6.2.3.2 Distribution

$x \in R^D$. Suppose $\Sigma \succ 0$.
$N(x|\mu, \Sigma) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$. Parameters: Σ, μ.

Reformulations $tr((x-\mu)^T \Sigma^{-1}(x-\mu)) = tr(\Sigma^{-1}(x-\mu)(x-\mu)^T)$.
Also, often writ as $\propto e^{-2^{-1}x^T P_i x + \mu_i^T P_i x} = e^{-x^T A x + b^T x}$. The mode-finding problem, given some sample points, is to find $\mu = A^{-1}b$, then: see Gaussian graphical model inference.

Singular covariance matrix If covariance matrix is singular, the expression is ill-defined. But, one can consider a low dimensional distribution embedded in a higher dimensional space.

6.2.3.3 Covariance matrix is symmetric

If Σ is a covariance matrix, it must be symmetric. As no complex numbers are invloved, Σ^{-1} in exponent can be taken to be symmetric; thence $\Sigma = \Sigma^T$ assumable.
If $\Sigma \succeq 0$: $\Sigma = U \Lambda U^*, |\Sigma|^{1/2} = \prod \lambda_i^{1/2}$.

6.2.3.4 Geometric view

Take level set; $N(x|\mu, \Sigma) = c$,
get $\Delta = (x-\mu)^T \Sigma^{-1}(x-\mu) = (x-\mu)^T U^* \Lambda^{-1} U(x-\mu) = c'$: hyper-ellipse, with u_i as major axes, $\lambda_i^{1/2}$ as radii, μ as center: see topology ref.
Δ is the Mahalonobis distance.

As product of univariate normal distribution So, take $y = U(x - \mu)$ as new axis. Then $N(x|\mu, \Sigma) =$
$\frac{1}{(2\pi)^{D/2}\prod \lambda_i^{1/2}} e^{-\frac{1}{2}\prod \frac{y_j^2}{\lambda_j}}$: thus factored into a product of univariate normal distribution variables.

A special case If (X_i) are $\perp$, Σ is diagonal, then this boils down to product of univariate normal distributions, as expected.

6.2.3.5 Product of normal distributions

$N(\mu_1, \Sigma_1 = P_1^{-1}) + N(\mu_2, \Sigma_2 = P_2^{-1}) = N(\mu, \Sigma)$ with $\Sigma^{-1} = P = (P_1 + P_2)$, $\mu = (\mu_1^T P_1 + \mu_2^T P_1)P^{-1}$: consider the form of the exponent: $-2^{-1}\sum_i x^T P_i x + \mu_i^T P_i x$.

6.2.4 ∞ dimensional Normal distribution

Aka Gaussian process. ∞ dimensional distribution, whose content in every finite dimensional subspace holds a finite dimensional Normal distribution. A gaussian distribution on functions.

6.2.5 Gaussian graphical models

6.2.5.1 Uncorrelated variables

Take the graphical model graph G corresponding to the multidimensional normal distribution. Take precision matrix $V = \Sigma^{-1}$. $V_{i,j} = 0$ corresponds to pairwise independence $X_i \perp X_j | X_{V-\{i,j\}}$ in G, and to the factorization $f_X(x) = \prod f_{i,j:V_{i,j}\neq 0}(x_i, x_j)$.
Inference in this graphical model is often interesting: it is equivalent to solving Ax = b for symmetric a.

6.3 1D distributions from exponential family

6.3.1 Polynomial rise with inverse exponential decay for largeness

Aka Gamma distribution gamma(a, b).
Models time to complete some task.
pdf $f_X(x) = \frac{b^{-a}x^{a-1}e^{-x/b}}{\Gamma(a)} \propto x^{a-1}e^{-x/b}$ for $x \in [0, \infty]$: Pf: using $\Gamma()$ defn. Mean $\mu = ab; var[X] = ab^2$ using $\Gamma()$ properties. By finding critical point of f(x), mode is b(a-1).
This is a single mode distribution.

Inverse Gamma distribution The distribution of U = 1/X, where $X \sim gamma(a, b)$.

This is again a single mode distribution.

6.3.1.1 Exponential decay distribution $expo(\mu)$

pdf: $f(x;m) = m^{-1}e^{-x/\mu}$ for $x \in [0,\infty]$. Same as $gamma(1,\mu)$.
CDF $F(x) = 1 - e^{-x/\mu}$ iff $x \geq 0$.
m is mean by integration by parts.
$var[X] = m^2$: use integration by parts to find $E[X^2]$; find $E[X^2] - m^2$.

Shift parameter Note that analogous distributions $f(x;t,m)$ can be defined with support $x \in [t,\infty]$ by adding a location parameter t.

Bilateral-exponential decay distribution Aka double-exponential decay distribution. Aka Laplace distribution.
pdf $f(x;t,m) = \frac{1}{2m}e^{-|x-t|/m}$.

The graph Mode is 0. You get an exponential tail. $\frac{1}{\mu}$ controls when the exponential decay kicks in. Aka exponential clock.

7 Other density families

7.1 Sampling distributions

Sampling distributions are distributions of the functions of samples drawn from other distributions.

7.1.1 Standard normal square sum

Aka Chi square distribution with k degrees of freedom.
If $X_i \sim N(0,1)$, $\sum_{i=1}^{k} X_i^2 \sim \chi_k^2$. This is same as the distribution of $\sum(\frac{Y_i-\mu}{\mu})^2$.
Used in goodness of fit tests. [**Check**]

7.1.2 Student's t distribution with k degrees of freedom

$\frac{Z}{\sqrt{W/n}}$, with $Z \sim N(0,1), W \sim \chi_n^2, Z \perp W$.

7.1.3 F distribution

$\frac{w_1/n_1}{w_2/n_2} \sim F_{n_1,n_2} : w_i \sim \chi_{n_i}^2$.

7.2 Heavy tailed distributions

$lt_{x\to\infty} \frac{Pr(X>x)}{e^{-\epsilon x}} = \infty$. Eg: Power law distribution, cauchy distribution.

7.2.1 Power law distributions

$p(x) \propto x^{-g}; p(x) = x^{-g}Z^{-1}$ for normalizing constant Z. $lt_{x\to 0}p(x) = \infty$: so must have lower bound x_{min}. log p vs log x graph looks like a straight line.
Aka scale free distribution. The only [**Find proof**]distribution with the property: $\exists g(b) : p(bx) = g(b)p(x)$.
A subset of heavy-tailed distribution family.
Includes Zipf's law distribution.

7.2.1.1 With exponential cutoff

$p(x) \propto x^{-\alpha}e^{-x\beta}$. log p vs log x graph looks like a straight line which suddenly bends: exponential term starts kicking in. Akin to gamma distribution.

7.2.1.2 Zipf's law for resource usage

Frequency/ probability of usage of resources often follows Zipf's law: Pr([res used]) $\propto f(resource)^{-k}$. Eg: words used in document.

7.3 Mixture distribution

Often, one models the pdf of X as being a convex combination of multiple pdf's.

7.4 Other pdf's

7.4.1 Uniform and triangular distributions

Uniform distribution; used when not information is available except min, max.
Triangular distribution is used when mode is also known.

7.4.2 Log normal distribution

Take $X \sim N(\mu, \sigma^2)$. Then $Y = e^X$ has log normal distribution. Wide variety of shapes, heavy tailed.

7.4.3 Gumbel distribution

Used in worst case analysis. CDF: $G(x|\mu, b) = e^{-e^{-\frac{x-\mu}{b}}}$, PDF: $g(x|\mu, b) = \frac{e^{-\frac{x-\mu}{b}}}{b}e^{-e^{-\frac{x-\mu}{b}}}$.

7.4.4 Probability simplex coordinate powering

Aka Dirichlet distribution. This is the conjugate prior for multinomial distribution.
Support is
$\left\{x \in R^k : \sum_i x_i = 1, x_i > 0\right\}$: or actually

$\left\{x \in R^{k-1} : \sum_i x_i < 1, x_i > 0\right\}$. pdf is $p(x;a) \propto \prod_{i=1:k} x_i^{a_i - 1}$ for parameters $a \geq 0$.

7.4.4.1 2-dim case

Aka beta(a,b) distribution. This takes up a wide variety of shapes: convex, concave, neither etc..
This is the conjugate prior for bernoulli/ binomial distribution - and a special case of Dirichlet distribution.
Pdf: $f(x) \propto x^{a-1}(1-x)^{b-1}$ for $x \in [0,1]$.

7.4.5 Wigner semicircle distribution

Supported on [-R, R], like a semicircle.

Part III

Model dependence among random variables

8 Distribution models

8.1 Discrete L: Response probability: Discriminative models

8.1.1 Boolean valued functions

One can use boolean valued functions to get deterministic models of the form $y = f(x)$. These functions are considered in the boolean functions survey and the computational learning theory survey.

8.1.2 Probability from regression models

Take any (continuous variable regression) model $f : X \to [0,1]$. Such a model can be interpreted as modeling the probability distribution f_L.

8.1.2.1 Advantages of modeling probability

The classifier doesn't care whether C_1 is called class 1 or class 100. So, better than solving regression problem with y as the target.

8.1.3 Model numeric labels with regression models

One may use regression models together with an appropriate round-off function to model discrete numerical labels.

8.1.3.1 Dependence on choice of ran(Y)

For the same k-classification problem, different choices of Y corresponding to $\{L_i\}$ can yield different models classifiers. Ideally they should be independent of choice of labels. So, logistic regression preferred.
Eg: For binary classification problem, picking $L_i = \{\pm 1\}$ yields different model from picking $L_i = \left\{\frac{N}{n_1}, -\frac{N}{n_2}\right\}$, which yields fisher's linear discriminant!

8.1.3.2 y in 1 of k binary encoding format

Make matrix X with rows $[1x_i^T]$. Make Y with rows y_i^T. Want to find parameters W such that $XW \approx Y$. Can try $\min_W \|XW - Y\|_F^2$, get solution: $(XX^T)\hat{W} = X^TY$. But $X\hat{W}$ can have -ve numbers which approximate Y; so not very desirable technique

8.1.4 Logistic model

Got k-class classification problem. Want to model class probabilities or log odds and make classification decision.

8.1.4.1 Log linear model for class probabilities

$\forall i \in [1:k] : Pr(C=i|x) \propto e^{w_{i0}+w_i^T x}$. So, $Pr(C=i|x) = \frac{e^{w_{i0}+w_i^T x}}{\sum_j e^{w_{j0}+w_j^T x}}$.
But this is over parametrized: The choice of w is constrained by the fact that specifying $Pr(C=i|x) \forall i = 1:k-1$ completely specifies the probability distribution.

8.1.4.2 Equivalent form: model log odds

$\forall i \in [1:k-1] : \log \frac{Pr(C=i|x)}{Pr(C=k|x)} = w_{i0} + w_i^T x$.
Get: $Pr(C=i|x) = \frac{e^{w_{i0}+w_i^T x}}{1+\sum_{j\neq k} e^{w_{j0}+w_j^T x}}$, $Pr(C=k|x) = \frac{1}{1+\sum_{j\neq k} e^{w_{i0}+w_i^T x}}$.
Same as the model described in previous subsubsection, with all $Pr(C=i)$ scaled to ensure that $Pr(C=i|x) \propto e^{w_{i0}+w_i^T x} Pr(C=k|x)$: done by ensuring that $w_k = 0$. Thus taking care of earlier overparametrization!

Symmetric notation Let $x \leftarrow (1,x), w_i \leftarrow (w_{i0}, w_i)$.

$$Pr(C=i|x) = \frac{e^{\sum_{c\in\{1,..m-1\}} w_c^T x I[c=i]}}{1+\sum_{j\neq k} e^{\sum_c w_c^T x I[c=j]}}$$

8.1.4.3 2-class case

For 2 class case, these are logistic sigmoid functions, thence the name.

8.1.4.4 Risk factors interpretation

$Pr(C_i|x)$ is modeled as a sigmoid function which $\to 0$ as $w_i^T x \to -\infty$ and $\to 1$ as $w_i^T x \to \infty$. So, can consider w_i as the vector of weights assigned to features $\{x_j\}$. $Sgn(w_i)$ usually indicates type of correlation with C_i, but could be reversed in order to compensate for weightage given to other features. Eg: C_i could be 'has heart disease', and features may be liquor, fat and tobacco consumption levels.

8.1.4.5 As a linear discriminant

Consider the binary classification case. Here, $\log \frac{Pr(C=1|x)}{1-Pr(C=1|x)} = w_0 + w^T x$. So, $w_0 + w^T x > 0 \equiv (Pr(c=1|x) > Pr(c=0|x))$

8.1.5 Estimating parameters

Given observations $(x^{(i)}, c^{(i)})$, find w to $\max_w Pr(c^{(i)}|x^{(i)}, w)$: maximum likelihood estimation.

8.1.5.1 Sparsity of model parameters

Sometimes, want w to be sparse or group sparse. In this case, for learning the parameters, lasso or group lasso is used.

8.2 Discrete L: Response probability: Generative models

8.2.1 Latent variable model

Assume that the parameter $W = w$ actually generates lower dimensional L, and that observation set X is generated from L using some stochastic transformation which is independent of w.
L is called the latent variable.

8.2.2 Assume conditional independence of input variables

Aka Naive Bayes. $Pr(L|\phi(X)) \propto Pr(L)Pr(\phi(X)|L) = Pr(L)\prod_i Pr(\phi_i(X)|L)$. $Pr(\phi(X)|L) = Pr(L)\prod_i Pr(\phi_i(X)|L)$ is the assumption. Model parameters $Pr(\phi_i(X)|L)$ and $Pr(L)$ are estimated from the training set $\{(X_i, L_i)\}$.
Co-clustering in a way recovers things lost due to the 'independence of probability of occurrence of features' assumption. [**Incomplete**]
One can conceive of a version of this classifier for the case where $L, \phi(X)$ are continuous. [**OP**]:

8.2.2.1 Linear separator in some feature space

The decision boundary can be specified by $\log Pr(l_1) + \sum_i \log Pr(\phi_i(x)|l_1) = \log Pr(l_2) + \sum_i \log Pr(\phi_i(x)|l_2)$.
Apply the following mapping for variables: $y_{i,d} = I[\phi_i(x) = d]$; and create a new set of parameters: $w_{i,d} = \log Pr(\phi_i(X) = d|l_1) - \log Pr(\phi_i(X) = d|l_2)$, and $w_0 = \log Pr(l_1) - \log Pr(l_2)$. Now, the decision boundary is just $w_0 + w^T y = 0$, which is a linear separator.

8.2.2.2 Success in practice.

Often works well in practice. Eg: In document classification.

8.2.2.3 Discriminative counterpart

Its discriminative counterpart is the class of all linear classifiers in a certain feature space, which corresponds to logistic regression. That, in general works better given a lot of samples.

8.2.3 Use exponential family models

8.2.3.1 Specification

For $ran(Y) = \pm 1$: Let $Pr(x|Y = i) \propto exp(\langle w_i, \phi(x)\rangle)$, and $Pr(Y = 1) = p$.
So, the corresponding discriminative classifier is: $Pr(y|x) = exp(\log(\frac{p}{1-p}) + \log(\frac{Z(w_0)}{Z(w_1)}) + \langle w_1 - w_0, \phi(x)\rangle)$, which is a linear classifier.
The corresponding discriminative classifier can be deduced directly using logistic regression.

8.2.3.2 Tree structure assumptions

In estimating, it is important to use the family of tree strucutred graphical models: We can't tractably compute $Z(w)$ otherwise. w_i can be done efficiently by computing the spanning tree of a graph among nodes with edges weighted by mutual information (Chow Liu algorithm).
Otherwise, mixture of trees are also used.

8.3 Latent variable models: Expectation Maximization (EM) alg

8.3.1 Problem

We have an observation $X = x$ and want to deduce the label Y.

8.3.1.1 Tough to Optimize likelihood

We want to $\max_w \log L(w|X = x) = \max_w \log \sum_y f_{X,Y|w}(x, y)$, but this expression often turns out to be hard to maximize due to non-convexity/ non-

smoothness. Suppose that this is the case. Also suppose that $f_{W|X,Y}$ is easy to maximize.
So, we resort to local optimization of a surrogate function starting from an initial guess of w.

8.3.1.2 Examples

May be want to find parameter w giving weights to a set of fixed Gaussians. Here, Y can be vector of id's of Gaussians whence observed data X comes from. A more common and important application is in estimating HMM parameters.

8.3.2 Iterative algorithm

Suppose that you are given $w^{(i)}$. We want to obtain $w^{(i+1)}$ such that $L(w^{(i+1)}) \geq L(w^i)$.

8.3.2.1 Intuition

Basic idea is to do the following repeatedly: at point $w^{(i)}$, to find a tractable and approximate surrogate $Q(w|w^{(i)})$ for $L(w|X)$, and maximize it to get a 'better' $w^{(i+1)}$.
Consider $Q(w|w^{(i)})$ from the E-step below. $Q(w|w^{(i)})$ is the expectation wrt $w^{(i)}$ over Y of the log likelihood of w given (X, Y). This seems to be a reasonable substitute for $L(w|X)$.

8.3.2.2 E-step

Take
$Q(w|w^{(i)}) = E_{y\sim w^{(i)}}[(\log f_{X,Y|w}(x, y))]$ to measure goodness of w in producing X and the current belief about Y.

8.3.2.3 M-step

Set $w^{(i+1)} = argmax_w Q(w|w^{(i)})$.

8.3.3 Analysis

8.3.3.1 Maximizing an approximation of the likelihood

Instead, construct a function Q(w) which lower bounds $\log L(w|X)$; then maximize it to get $w^{(i+1)}$; repeat.

8.3.3.2 Q(w) is a lower bound

Q(w) a lower bound for $\log L(w|x)$. [**Proof**]: Regardless of how $Y \sim w^{(i)}$ is distributed, $Q(w) = E_y \log L(w|x, y) \leq \log L(w|x)$ because $E_t \log t \leq \log \max_{t\in T} t \leq \log \sum_T t$. □

8.3.3.3 Convergence

Q() lower bounds L(), but we cannot guarantee that the $\max_w Q()$ does not lead us away from the local maximum. So, monotonic convergence is not guaranteed. [**Check**]

9 Graphical models

9.1 Graphical model G of distribution

9.1.1 The modeling problem

Got RV's $X = (X_i)$, $f_X(x)$: joint probability density. RV's as nodes. Edges representing dependencies.

9.1.1.1 Distribution structure/ sparsity

Seek to represent some factorization of the joint probability distribution concisely, thence conditional independence relationships too. In many cases, these factors involve small subsets of variables: sparsity in the dependency graph.
Eg: $f_X(x) = Z^{-1} \prod_{C \subseteq V} \phi_C(x_C)$. Compare notation with exponential family distributions.

Graphical model family A graph alone describes conditional independence relationships which is satisfied by many distributions.

9.1.1.2 Uses

Any distribution can be represented by a (maybe complete) graphical model, but it becomes interesting only when the graph/ model is sparse.
Useful in representing causal relationships.
The factorization of Pr(x) lets ye store the joint probability distribution very concisely: usually ye would need $ran(X_i)^n$ space.
Can do fast inference using graph theoretic algs.
Can characterize running time and inference error bound in terms of properties of the underlying graph.

9.1.2 Factor graphs

9.1.2.1 Factors of Pr(x)

Bipartite graph of shaded ovals ($\{i\}$ for factors $f_i(\Gamma(i))$: any nonnegative fns) and ovals (RV's $\{X_i\}$). $f_X(x) = Z^{-1} \prod f_i(\Gamma(i))$. This is a 'hypergraph' among $\{X_i\}$, with generalized edges connecting 2 sets of variables.
Also defines another graph, Γ relationship amongst $\{X_i\}$; and thence 'path' is defined.

Connection with exponential families Same as in the undirected model case.

9.1.2.2 Conditional independence

If every path between RV's X, Y passes through Z, Z separates X, Y. If Z separates X, Y $X \perp Y|Z$: Pf: See undirected model case.
So, can think of Z as an observed variable, Z blocks flow of information from X to Y. So, $\Gamma(X)$ is its Markov blanket.

9.1.2.3 Expressiveness

Can express any factorization. Eg: can design factor f(X, Y) to say that X and Y are ϵ apart; so factors called compatibility functions.

9.1.3 Undirected graphical models

Aka Markov random field.

9.1.3.1 Factorization

$f_X(x) = Z^{-1} \prod \phi_{C_j}(x_{C_j})$, where C_j are cliques of various sizes in G.

As an exponential distribution family See section on exponential families.

9.1.3.2 Conditional independence properties

Aka Markov properties. Conditional independence properties, markov blanket same as in Factor graphs.

Global Markov Take any A, B, Z. If Z separates A and B, $A \perp B|Z$.
[**Proof**]: Factorization implies this. Take A, B, Z; expand A and B to get A', B' which include all nodes reachable from A and B without crossing Z; So, $f_X(x) = f(x_{A'}, x_Z)f(x_{B'}, x_Z)$; so $A' \perp B'|Z$. □

Local Markov $X_i \perp X_{V-i-N(i)}|X_{N(i)}$. Global markov implies this.

Pairwise Markov If $(i, j) \notin E$, $X_i \perp X_j|X_{V-\{i,j\}}$. Implied by Global Markov. Local Markov implies this.

Factorization from pairwise markov for many Pr(x) (Hammersley Clifford) If $\forall x : f_X(x) > 0$ pairwise markov implies factorization.

9.1.3.3 Tree structured case

Importance It is easy to compute the partition function for this case. There exist efficient algorithms to do inference accurately on such models, and there are efficient algorithms to find the closest tree structred graphical model to any distribution.

Form and connections $f_X(x) \propto \prod_{(i,j)\in T} \phi_{i,j}(x_i, x_j)$.

As directed model Now, consider any node, say x_1, to be the root of the tree.
$f_X(x_1, x_{\Gamma(1)}) \propto \prod_{j\in\Gamma(1)} \phi_{1,j}(x_1, x_j)$. But,
$f_X(x_1, x_{\Gamma(1)}) = f_{X_1}(x_1) \prod_{j\in\Gamma(1)} f_{X_j|X_1}(x_j|x_1)$ from the conditional independence property of undirected graphical models. Applying this procedure recursively, one gets a directed graphical model.

In terms of marginals Consider the corresponding directed model.
$f_{X_1,\Gamma(1)}(x_1, x_{\Gamma(1)}) = f_{X_1}(x_1) \prod_{j\in\Gamma(1)} f_{X_j|X_1=x_1}(x_j)$
$= f_{X_1}(x_1) \prod_{j\in\Gamma(1)} f_{X_j}(x_j) \prod_{j\in\Gamma(1)} \frac{f_{X_j,X_1}(x_j x_1)}{f_{X_j}(x_j) f_{X_1}(x_1)}$.
Applying this procedure repeatedly, we get:
$f_X(x) = \prod f_{X_i}(x_i) \prod_{(i,j)\in E} \frac{f_{X_i,X_j}(x_i,x_j)}{f_{X_i}(x_i) f_{X_j}(x_j)}$.

9.1.3.4 Pairwise graphical model

A subclass. $f_X(x) \propto \prod_i \phi_i(x_i) \prod_{(i,j)\in E} \phi_{i,j}(x_i, x_j)$.

9.1.3.5 Hierarchical models

A factor $\phi_c(x_c)$ exists only if, for all $s \subset c$, a factor $\phi_s(x_s)$ exists.

9.1.3.6 Discrete models

$\{dom(X_i)\}$ are discrete.

Pairwise-ification of discrete models Can add some extra variables, rewrite with all C_j being pairwise: If cliques of size $p' > 2$ exist, collapse that clique into a single node, expand the state space.
Note that, just because you know how to learn pairwise graphical models, you cannot simply construct a general discrete model learning algorithm: you don't know which nodes to collapse.

The general form Consider exponential family attached to discrete graphical model G of n vars. Let $|dom(X_i)| = |M| = m$. Can assume G is pairwise.

We can completely specify $\phi_{i,j}(x_i, x_j)$ by parameter matrix $T_{i,j}$ with $T_{i,j,k,l} = \phi_{i,j}(k, l)$.

$$f_X(x) =\propto e^{\sum_{(i,j)\in E} T_{i,j,x_i,x_j}} = e^{\sum_{(i,j)\in V^2} \sum_{k,l\in M^2} T_{i,j,k,l} I[x_i=k] I[x_j=l]}$$

. We can think of this as an exponential family distribution involving $|V|^2 m^2$ auxiliary features/ covariates $y_{ij} = I[x_i = k]I[x_j = l]$. But this distribution $f_X(x)$ is now overparametrized, as $y_{i,j}$ are not linearly independent. [See section on minimal parametrization of exponential family distributions.]
Let $M' = M - \{m\}$. Using a minimal parametrization, we get an exponential family distribution involving only features $y_{ij;kl\in(M')^2} = I[x_i = k]I[x_j = l]$ and $y_{i;k\in(M')} = I[x_i = k]$. So, $Pr(X = x \in M') \propto exp(\sum t_{i;k} y_{i;k} + \sum t_{ij;kl} y_{ij;kl})$.

Ising model $Pr(X = x|t) \propto e^{\sum_i t_i x_i + \sum_{(i,j)\in V^2} t_{i,j} x_i x_j}; dom(X_i) \in \pm 1$. Any binary undirected graphical model involving variables Y_i with range $\{1, 2\}$ can be expressed like this: just consider the minimal parametrization of such distribution using the auxiliary features described earlier.
For signed edge recovery for the class of Ising models given a few observations, see structure learning part in statistics ref. Originally used in physics to model electron spins' interactions in the case of magnetism.

9.1.4 Junction tree model

Take an undirected graph G, find a junction tree T for it (see graph theory ref). Belief propagation algorithms work well over trees; hence this. Like a factor graph, there are 2 types of nodes: a set of nodes for cliques C in G. They are connected to each other through separators S.

9.1.4.1 Factorization

$f_X(x) = \frac{\prod_{c\in C} f_{X_c}(x_c)}{\prod_{s\in S} f_{X_s}(x_s)^{|\Gamma(s)|-1}}$.

9.1.5 Directed

Aka Bayesian networks; but needn't be learned using Bayesian methods.

9.1.5.1 Extra notation

Shorthand for N nodes with identical parentage: a plate annotated by N, with a single node inside. Can represent deterministic variables with solid dots. Can represent observed variables as shaded nodes.

9.1.5.2 Factorization

Every X_i annotated with $f_{X_i|par(X_i)}$ (aka factors).
$f_X = \prod_i f_{X_i|par(X_i)} = \prod_i f(X_i, par(X_i))$.

There are many bayesian networks to represent f_X based on different decompositions: eg: $X_1 = X_2 + X_3$. Not all are equally concise. Concise when expressing causal relationships.

Undirected 'moralized' graph Make all edges undirected, but 'marry off' all unmarried parents: make cliques involving child and parents. These graphs are equivalent in terms of conditional independence.

9.1.5.3 Marginal independence

If X, Y don't have a common ancestor: $X \perp Y$. But, conditional independence, $X \perp Y|E$ need not hold if E has a common child of X and Y; Eg: $X_1 = X_2 + X_3$.

9.1.5.4 Dependency seperation of X, Y by Z

Aka d-separation. Every undirected path (X, Y) blocked by $W \in Z$. 2 types of blocking: $\rightarrow W \leftarrow$: W not given; $\rightarrow W \rightarrow$ or $\leftarrow W \rightarrow$: W given.
d-separation is graph-independent: Even when multiple graphs model same distribution, the conditional independence relationship deduced from any of them hold.

Global Markov property Let A, B, Z be sets of variables. $A \perp B|Z$ for all d-separating Z. Thence, markov boundary of X is
$\{par(A), chi(A), par(chi(A))\}$. Implied by factorization.
Also, if S separates A, B in moralized graph, $A \perp B|S$. But, when S does not separate A, B: look at subgraph of A, B, S.

Check d-separation Use breadth-first-search to find unblocked paths. Aka Bayes ball algorithm.

9.1.5.5 Other conditional independence properties

Local markov property desc(i) :=descendents of i. $X_i \perp X_{j \notin desc(i)}|par(i)$.

Pairwise markov property $X_i \perp X_j|X_{\Gamma(i)-j}$.

Connections Factorization $\equiv$ Global Markov $\equiv$ Local Markov $\implies$ Pairwise markov. If $f_X(x)$ has full support, pairwise markov $\implies$ local markov.

9.1.5.6 Marginalized DAG

Let G = (V, E) be the DAG corresponding to Pr(x). The DAG corresponding to $f_{X_{V-A}}(x_{V-A})$ is obtained as follows: Take subgraph S in G induced by (V - A). For every $(u, v) \in (V - A)^2$, add a new edge if $\exists$ a directed path (u, s, v) in G, such that s is a sequence of vertices in A. Proof: Using factorization.

9.1.6 Comparison

9.1.6.1 Expressiveness

Take rain, sprinkler, grass wet (R, S, G) causal model. $R \perp S$ but $R \not\perp S|G$: 'Explaining away' phenomenon. Can't express this with other models.
Take rectangle shaped undirected graph. Can't make equivalent directed graph.
Undirected graphical models are better at expressing non-causal, soft relationships amongst RV's. Directed models are usually very intuitive to construct.
Undirected models less expressive than factor graphs. [**Find proof**]

9.1.6.2 Structural equivalence

Tree structured undirected graphical model can be expressed as a directed graphical model with the same structure: this is detailed in the undirected graphical models section.
The reverse is not true: as seen from the rain, sprinkler, grass wet example. But if edges in the DAG do not meet, a tree structured directed graphical model can be expressed as an undirected graphical model.

9.1.6.3 Independence relationships amongst vars

Conditional independence easier to determine in undirected models compared with directed graphical models. But marginal independence easier to determine in the former.

9.2 Inference, decoding using Graphical model

See statistics survey for the following: structure learning (learn graph from data); parameter learning (learn parameters given graph).

9.2.1 Problems

Consider some $S \subseteq V$.

9.2.1.1 Inference problems

Find marginals f_{X_S}, or the partition function Z, or find $E[g(x)]$, when $g(x)$ factors nicely according to the graphical model.

9.2.1.2 Decoding problem

Find component $\hat{x}_S$ of the mode $\hat{x} = \arg\max_x f_X$.

Global maximum vs marginal maxima Note that this is different from the marginal maximum $\arg\max_{x_S} f_{X_S}$ which may be found by solving the inference problem to find f_{X_S} and then taking its maximum.

One cannot simply find the local/ marginal maxima $\arg\max_{x_S} f_{X_S}$ and use it to find the global maximum.

9.2.1.3 Evidence

Maybe you want to solve these problems when values for some variables may be fixed: Eg: $X_T = x_T$.

9.2.1.4 Solving for all variables

Another variation to the inference and decoding problems is to solve them for all sets of the form $S = \{i \in V\}$.

9.2.2 Factorization and graph-based computations

9.2.2.1 Benefit of factorization

Inference problems involve summation over a subset $ran(X)$, while decoding problem involve finding the maximum over it.
Suppose that $f_X(x) = Z^{-1} \prod_{c \subseteq V} \phi_c(x_c)$. Distributive law is the key to summing/ maxing this function efficiently. Eg: see elimination algorithm, junction tree algorithm.

Elimination ordering Order the factors to get $f(x) = \phi_{c_1}(x)...$ Now, if you have variables X_T involved only in factors $\geq i$, you get: $\sum_{x_{V-S}} f_X(x) = \sum_{x_{V-S-x_T}} \phi_{c_S}(x) .. \sum_{x_T} \phi_{c_i}(x)...$ An identical ordering is useful if we were doing $\arg\max_{x_{V-S}} f(x)$ instead.
This yields us the following reduction in dimensionality.

Reduction in dimensionality Suppose that $\max_i ran(X_i) = D$. Without the elimination ordering, we would have had to consider a set of $D^{|V-S|}$ values during summing/ maxing. As a result of using the elimination ordering, we now consider a set of $D^{|V-S-T|} + D^{|T|}$ values to do the same.
Thus, using this trick repeatedly, suppose that we find the elimination ordering $f_X(x) = \prod_{c \in p(V)} \phi_c(x_c)$ where $p(V)$ is a partition of V. Then, we will be only be summing/ maxing over $|p(V)| D^{\max_{c \in p(V)} |c|} = O(D^{\max_{c \in p(V)} |c|})$ values.

Finding the right order Not all orders are equally good. There is a natural way to get this ordering for trees: consider the elimination algorithm.

9.2.2.2 Graph traversal view

Try to model the problem as one of making special graph traversals. Try to use local computations to replace global computations. Can think of this as nodes passing messages to each other.

9.2.3 Belief propagation

9.2.3.1 The Bottom-up idea

Exploit factorization and the elimination ordering we can solve the problem bottom-up.
If you are finding $\arg\max_x f_X(x)$, this is the max-product algorithm, if finding $f_{X_1}(x_1)$, it is the sum product algorithm.
The idea: take local belief, take max or sum, propagate it to other nodes which need this to calculate their belief.

9.2.3.2 Node Elimination algorithm: Undirected Trees

Remove nodes to sum out/ max out one by one. Suppose want to find $\arg\max f_{X_{V-1}|X_1=x_1}(x_{V-1})$. Then, root the tree at x_1, and do the following. Message, a definition of a function, every node X_j tells its parent X_i: $m_{j\to i}(x_i) = \max_j \phi_{i,j}(x_i, x_j) \prod_{k\neq i,(j,k)\in E} m_{k\to j}(x_j)$. This is the message passing implementation. Belief of x_1: $b_1(x_1) = \prod_{(j,1)\in E} m_{k\to 1}(x_1)$.
Can use similar algorithm to find marginal $f_{X_1}(x_1)$.

Using known values Suppose $X_2 = x_2$ is fixed in the above process. Nothing changes in the algorithm itself - only X_2 is thought of having only one value in its range while being summed over/ maxed over etc..

Finding conditional marginals Suppose you want to find $\arg\max Pr(x_{V-1}|x_1, x_2)$. Root the tree at say x_1, execute the algorithm as usual; when you encounter factors involving x_2, don't sum/ max over x_2.

9.2.3.3 Reusing messages: Undirected Tree

Maybe you want to find $f_{X_i}(x_i)\forall i$, and want to reuse messages (computations involving summing out). Then, simply use these update rules: $m_{j\to i}(x_i) = \max_j \phi_{i,j}(x_i, x_j) \prod_{k\neq i,(j,k)\in E} m_{k\to j}(x_j)\forall i, j$.
The algorithm is naturally distributed - so scales well with number of nodes. Also, there is no need to compute an elimination ordering.

Feasibility Each message depends on certain other messages, and it is computed when these messages are available. This is always possible for trees as there are no cyclical dependencies, and all messages are computed eventually.

9.2.3.4 Tree Factor graphs

Want to find $f_{X_i}(x_i)\forall i$, given tree structured factor graph. Root it at x_1, for every variable v and factor f, use the update rules:
$m_{v\to f}(x_v) = \prod_{f'\neq f} m_{f'\to v}(x_v), m_{f\to v}(x_v) = \sum v' \neq v f(x) \prod_{v'\neq v} m_{v'\to v}(x_{v'})$.
Belief $b_v(x_v) = \prod_f m_{f\to v}(x_v)$.

9.2.3.5 General undirected graphs

Use junction trees Maybe you don't have a tree, but a graph for which you can get a junction tree. If the graph does not have one, can always convert it to a chordal graph by adding edges: but finding the minimal chordal supergraph is NP hard.
Belief propagation proceeds as in tree factor graphs, except the clique nodes play the role of variables, and the separators/ nodes representing variables shared between cliques play the role of factors.
Belief for each clique C is thus easily calculated; by induction over number of cliques in the clique tree, $b_c(x_c) = \sum_{V-c} f_X(x)$, as expected.
Finding the belief for each variable depends exponentially on tree width: must consider $dom(X_i)^{|C|}$ values while summing/ maxing.

Tree reweighted max product Take $p(x) \propto g_1(x)g_2(x)$, such that every clique involved in p(x) is either in g_1 or in g_2. Then, if $x^* \in \arg\max g_1(x) \wedge x^* \in \arg\max g_2(x), x \in \arg\max p(x)$. So, can find smart ways of splitting p; then maximize each g; if the intersection of the maximal points is not empty, then done; otherwise, move around edge-mass.

9.2.3.6 Approximate inference: Loopy belief propagation

Trouble because of loops Suppose there were loops, you can try initializing all incoming messages at all nodes with 1, applying update rule at each node repeatedly. Each node calculates $m_{i\to j}^{(t+1)} = \phi_{i,j}(x_i, x_j) \prod_{k\neq j} m_{k\to i})^{(t)}$. Also, as messages (a function output vector $m(x_i)$) may keep growing bigger; may need to normalize each message at each iteration.

Applicability There are almost no theoretical guarantees of convergence or correctness; but widely applied in practice. But, when applied to trees, it is consistent with usual belief propagation, and yields the right answer.
An example in case of the inference problem involved in decoding binary linear codes is given in the information/ coding theory survey.

Computation tree at iteration j wrt node i A way to visualize the undirected graph, as it looks to node i at iteration j, while it calculates the belief $b_i^{(j)}(x_i)$ during loopy belief propagation. For j=0, the tree is just the single node i. For every j, you add a level to the tree, indicating new messages which are considered in $b_i^{(j)}(x_i)$ - the new leaves attached to a node k are $\Gamma(k) - par(i)$. Each tree is a valid graphical model in itself.
Eg: consider triangle 1, 2, 3. Initially, tree is 1. Then new level (2, 3) is added. Then children 3', 1' are added to 2; and 1', 2' are added to 3. These copies of nodes are conceptually different from the original: the messages they send are different.

'Correctness in case of steady state' results This is useful because: maybe loopy belief propagation will be in a steady state, before there is a loop in the computation tree - so highly dependent on the initialization!

Damped max product Use control theory idea to force oscillating system towards a steady state. Each node actually uses message $m_{i\to j}^{\prime(t+1)} = m_{i\to j}^{(t)l} m_{i\to j}^{(t+1)(1-l)}$.

Max-product on single cycle But, if you are doing max-product on a graph which is a single cycle, and if you hit a steady state for all messages, then the computation yields the right answer to $\arg\max_x f_{|X_1=x_1}(x$. This also holds for graphs which are trees, single cycles or single cycled trees.
Proof idea: Consider the computation tree T wrt x_1 at the steady state. Then, belief computed at x_1 corresponds to $\arg\max_x f_{X\sim T}(x_{V-1}|x_1)$, the max product belief correct for this tree. But, see that
$Pr_T(x_{V-1}|x_1) = tPr(x_{V-1}|x_1)^k$ for some k, t. Thence relate argmax Pr_T with argmax Pr.

9.2.4 For Gaussian graphical models

Maybe given normal distribution of in the form $f_X(x) \propto e^{-2^{-1}x^T Px+hx}$ by specifying P and $h = P\mu$, where P is the precision matrix. For every $P_{i,j} \neq 0$, there is an edge in the model graph G.
Max product finds the mean; sum product finds the marginal: either case reduces to finding the mean μ; so they correspond to executing the same algorithm. Marginalizing or maxing over a gaussian distribution yields another gaussian $e^{-2^{-1}x^T P'x+h'x}$, so the messages passed during message passing algorithm correspond to the parameters of this expression.

9.2.4.1 Connection with solving Ax = b

Essentially solving $P\mu = h$ for μ; perhaps loopy belief propagation can be used to solve Ax = b for very large $A \succeq 0$. Convergence happens only if P is diagonally dominant.
If G is a tree, then this corresponds to Gaussian elimination.

9.2.5 Directed graphical models

One can simply convert directed graphical models to equivalent undirected models and use inference algorithms described for them.

10 Sparse signal detection

10.1 Scale mixture models

[**Incomplete**]

11 Affinity modeling

11.1 Problem

One wants to probabilitically model 'affinities' (joint, conditional probabilities) of entities of two or more types. Entity types are modeled by discrete random variables (say W and D).

11.1.1 Motivation

Besides common motivations for modeling joint distributions of random variables, one may want to model affinities probabilistically in order to get low dimensional representations of one or both of these entities (motivations for which are described in the dimensionality reduction chapter of the statistics survey).

11.2 Non probabilistic ways

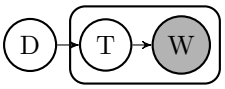
These are considered in the latent factor analysis section in the dimensionality reduction chapter of the statistics survey.
Eg: Latent Semantic Analysis (LSA), aka Latent Semantic Indexing (LSI): Use SVD to get factors for documents and words.

11.3 pLSA

Probabilistic LSA.

11.3.1 Aspect model

Each document is a convex combination/ mixture of topics, each topic defines a distribution over words; each word is drawn from this mixture of distributions. $Pr(w|d) = \sum_t Pr(t|d)Pr(w|t)$. So, $Pr(w,d) = Pr(d)\sum_t Pr(t|d)Pr(w|t) = Pr(t)\sum_t Pr(d|t)Pr(w|t)$: observe 2 factorizations.



11.3.2 Modeling assumptions

Bag of words assumption: given topic, words are chosen independently. Conditional independence: Given a mixture of topics (d), $w_1|t \perp w_2|t$.

11.3.3 Dimensionality reduction

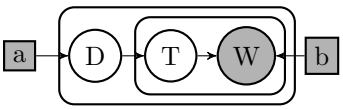
Each document, which was earlier a vector in the vocabulary space, is now a vector in the topic space.

11.3.4 Defects

Unclear how to assign probability to unseen item.

11.4 Latent Dirichlet Allocation (LDA)

Attempt to model observed bags of words at the corpus level. Look upon documents in corpus as having been generated by a process parametrized by corpus-level constant a. Also, add corpus level constant parameter b as extra parameter for generating words, given a topic.



12 Modeling stochastic processes

12.1 Stochastic process with state space T

Aka random process. T-valued random variable/ state sequence indexed by $r \in R$ (often time): visualize as a time-series - a directed graph which is a straight line.

12.1.1 Multiple coin toss processes

Consider a sequence of coin tosses. Let X_i model the outcome of the i-th toss. Bernoulli process: iid bernoulli trials: the same coin is tossed multiple times, that is $\forall i : X_i \sim p$. Resulting from such a process is the Binomial distribution for $\sum_i X_i$.
Poisson trials: independent but not necessarily identically distributed trials, that is $X_i \sim p_i$.

12.1.2 Continuous time

Aka flow. [**Incomplete**]

12.2 State transitions

Many models often propose that behind the production of a sequence of observations, there are (possibly hidden/ latent) changes in internal state. They allow transition from one state to another to be stochastic.
Let the state at time t be X_t. Let V be the set of possible states.

12.2.1 Assumptions about state transitions

12.2.1.1 Dependence solely on prior state

A model which assumes the Markov property described below is often convenient to represent states.

Sequence distribution: chain structure Markov property/ assumption: Future states are independent of past states: $f_{X_{t+1}|X_t..X_0} = f_{X_{t+1}|X_t}$.
A state transition model with this property is called a Markov chain/ bigram state chain, considering the chain-like graphical model of the distribution of the variables $\{X_t\}$.
So, the set of distributions $\left\{f_{X_t|X_{t-1}=t} \forall t \in T\right\}$ completely describes the state transition model.

12.2.1.2 Dependence on prior k states

Suppose that the concept of a bigram state chain is generalized to a k+1-gram state chain. So, $f_{X_t|X_{t-1}..X_0} = f_{X_t|X_{t-1}..X_{t-k}}$: a weaker independence property holds.

12.2.1.3 Reduction to bigram state chain

Consider a bigram state chain with the state sequence being (Y_t), and $ran(Y_t) = T^k$. Then, we will have the bigram Makrov property: $f_{Y_t|Y_{t-1}..Y_0} = f_{Y_t|Y_{t-1}}$.
Since it is easy to translate between (Y_t) and (X_t), we have a way to do learn k-gram model using corresponding algorithms for the bigram model.

12.2.2 Describing bigram model

12.2.2.1 State transition matrix

Thence get a $|V| \times |V|$ transition matrix P with
$P_{x,y} = f_{X_{t+1}|X_t=x}(y)$; also see stochastic matrices in linear algebra ref.
Probability distribution vector over states at r: p_t. $p_t = Pp_{t-1} = P^t p_{t-1}$.

12.2.2.2 State transition graph

Consider state graph G = (V, E) with transition probabilities on edges, labeled with transition probabilities independent of time (time homogenous). This labeled graph is a diagrammatic way of accurately representing a markov chain.

12.2.2.3 Types of states and chains

Recurrent state: $Pr(revisit) \to 1$. Aperiodic state: $GCD\left\{t : P^t_{x,x} > 0\right\} = 1$.
Irreducible: No unreachable state. If finite and irreducible, +ve recurrent. If +ve recurrent et aperiodic: ergodic.

Detailed balance property Reversible chain: $\exists \pi : \forall x, y : \pi_x P_{x,y} = \pi_y P_{y,x}$.

12.2.2.4 Learning transition probabilities

Given a sufficiently long sequence X_i, one can estimate various transition probabilities $P_{x,y}$ by $\frac{\sum_t I(X_t=y|X_{t-1}=x)}{\sum_t I(X_{t-1}=x)}$.

12.2.3 Unique Stationary distribution π of ergodic chains

$\forall x, y :$
$lt_{t\to\infty} P^t_{x,y} = \pi_y$. Find π: $P\pi = \pi$, $\sum \pi_i = 1$; or inflow = outflow. If time-reversible, π uniform.

12.2.4 Mixing time of Ergodic chain

12.2.4.1 Purpose, definition

Suppose that given the state transition graph of a markov chain, one wishes to sample a state from the stationary distribution.
One way to do this would be to do a long random walk (see randomized algorithms survey) on the state transition graph.
The mixing time of a Markov chain is the time taken for this sampling process to lead to a distribution close to the stationary distribution.
Mixing time also determines number of transitions to make before you can take a sample roughly independent from previous sample.

12.2.4.2 Coupling lemma

Start 2 identical copies of markov chain starting from arbitrary states. States at time T: X_T, Y_T. $Pr(X_T \neq Y_T | X_0 \neq Y_0) \leq \epsilon \Rightarrow t(\epsilon) \leq T$. Variation distance is non-increasing.
Let $\sum$ smallest column entries = m. Then, $||p^t_x - \pi|| \leq (1-m)^t$. $t(\epsilon) \leq t(c)(\ln \epsilon / \ln c)$.

12.2.4.3 Mixing time bound

Select clever coupling, maybe define distance function d_t and show that $Pr(d_t \geq 1) = E[d_{t+1}|d_t] \leq bd_t$ for $b < 1$, bound prob that chains haven't converged, use coupling lemma. Maybe take 2-step chain, use geometric coupling.

12.2.5 Straight line state transitions

12.2.5.1 Gambler's winnings

Suppose that two gamblers start with seed money: l_1, l_2. They toss a fair coin and bet a dollar until one of them is bankrupt.
From the perspective of player 1, this can be modeled as a markov chain with the state space representing the amount of money player 1 has: ranging $\{0, ..l_1 + l_2\}$. The initial state of the player is l_1, and transition probabilities are defined thus: $Pr(X_t = l_k + 1|X_{t-1} = l_k) = Pr(X_t = l_k - 1|X_{t-1} = l_k) = 1/2$ for $k \in 2..l_1 + l_2 - 1$, with 0 and $l_1 + l_2$ being terminal states.

Analysis using martingale property $E[X_t] = X_{t-1}$; and so $E[X_\infty] = l_1$. So $Pr(X_\infty = l_2 + l_1)(l_2 + l_1) = l_1$. So, Pr(1 wins)=$\frac{l_1}{(l_1+l_2)}$.

12.2.5.2 Queue

Arrival rate a, departure rate m. $\pi_i = (\frac{a}{m})^i(1 - \frac{a}{m})$. $h_{u,v}$ = E[Steps from u to v]. $\pi_i = 1/h_{i,i}$.

12.3 Martingale (Z_n) wrt filtration

12.3.1 Problem

Suppose that one observed RV (Z_n) and a filtration or a series of events (F_n), with the property that $F_n \supseteq F_{n-1}$.
Suppose further that:
$E[|Z_n|] < f(n) < \infty$, that Z_n is fully determined when F_n is observed, and $E[Z_n|F_{n-1}] = Z_{n-1}$ (or $E[Z_n|F_{n-1}] - Z_{n-1} = 0$).
This process is the martingale (Z_n) wrt filtration (F_n).

12.3.1.1 Example

The filtration can correspond to the observation of a sequence of random variables $(() X_n)$.
Note that this defines martingale (X_n) wrt itself. Eg: Wealth after 100 fair-coin-toss bets, Brownian motion.

12.3.2 Properties

Note that this implies that $E[Z_n] = E[Z_0]$.

12.3.3 Stopping time T

One can stop the stochastic process based on past (not future) bets/ Observations of X_i; the corresponding time is called the stopping time.
Stopping theorem: If $E[T] < \infty$ or T bounded or $|Z_i| < c$, then $E[Z_T] = E[Z_0]$. Wald: If X_i iid, T stopping time: $E[\sum_{i=0}^{T} X_i] = E[T]E[X]$.

12.3.4 Doob martingale

Anything like $Z_i = E_{X_{i+1..X_n}}[f(X_1..X_n)|X_1..X_i]$ fits defn of Martingale:
Eg:
$E_{X_2..}[Z_2|X_1] = E_{X_2..}[E_{X_3..}[f(X_i)|X_1, X_2]|X_1] =$
$\sum_{x_2} E_{X_3..}[f(X_i)|X_1, X_2 = x_2]Pr_{X_2}(X_2 = x_2|X_1) = E_{X_2..}[f(X_i)|X_1] = Z_1$.

12.3.5 Find expected running time of a game

Make a martingale, use Wald's equation.

12.3.6 Concentration around starting value

(Azuma) For martingale $\{X_i\}$:

$|X_k - X_{k-1}| < c_k : Pr(X_t - X_0 \geq l) \leq e^{-\frac{l^2}{2\sum c_k^2}}$.
Eg: If you make small bets then you stay near mean.
[**Proof**]: Define new RV: $Y = X_t - X_0 = \sum Y_i, Pr(e^{aY} \geq e^{al}) \leq \frac{E[e^{aY}]}{e^{al}} = \frac{E[\prod e^{aY_i}]}{e^{al}} = \frac{\prod E[e^{aY_i}]}{e^{al}}$ (from independence of $\{Y_i\}$). Take $a > 0$.
As e^{aY_i} is convex and $Y_i \in [-c_i, c_i]$, so $e^{aY_i} \leq \frac{e^{ac_i}(1-\frac{Y_i}{c_i}) + e^{-ac_i}(1+\frac{Y_i}{c_i})}{2} \leq \frac{e^{ac_i}+e^{-ac_i}}{2}$ as $e^{ac_i} > e^{-ac_i}$. So $E[e^{aY_i}|X_1..X_i] \leq \frac{e^{ac_i}+e^{-ac_i}}{2} \leq e^{(ac_i)^2/2}$ from e^x series.
So, $Pr(e^{aY} \geq e^{al}) \leq e^{-al}e^{\sum_i (ac_i)^2/2}$. Setting $a = \frac{l}{\sum c_i^2}$, we get the result. □
Core Idea: In the foregoing proof, the crucial idea was considering the exponentiated event, which could then be decomposed and bounded due to independence. The algebraic trickery in selecting the right value for a and in coming up with the bounds were interesting. ‖

Applying to martingale $\{-X_i\}$: $Pr(X_t - X_0 \leq -l) \leq e^{-\frac{l^2}{2\sum c_k^2}}$.

12.3.6.1 Applied to Doob Martingale

$Z_i = E[f(X_1..X_n)|X_1..X_i]$. If f satisfies Lipschitz condition with bound c (max change c in $f(X_1..X_n)$ when X_i changes): $Pr(|E[f(X_1..X_n)] - f(X_1..X_n)| \geq l) \leq 2e^{-\frac{l^2}{2nc^2}}$. Aka method of bounded differences (MOBD).
Note: No independence needed till here.

12.3.6.2 Additive Bound for deviation from mean

(Azuma Hoeffding) So, let independent, not necessarily identically distributed $X_i \in [b, c]$, $f(X_1..X_n) = X = \sum X_i$. $Pr(|\sum X_i - \sum \mu_i| \geq na) \leq e^{-\frac{n^2a^2}{2nc^2}}$.

Application in estimating mean If $\{X_i\}$ also identically distributed: $Pr(|X - n\mu| \geq na) \leq e^{-\frac{n^2a^2}{2nc^2}}$.

12.3.6.3 Additive deviation bound for sum of Poisson trial RV's

If $\frac{X}{n} = \hat{p}, Pr(|\hat{p} - \mu| \geq \epsilon) \leq 2e^{-\frac{n\epsilon^2}{2}}$. $1 - \epsilon$ confidence interval for parameter p.

12.4 n-gram model

12.4.1 Model

12.4.1.1 Subsequence/ prefix probabilities: notation

First, one models the probability $Pr(w_n|w_{1:n-1})$ of a word w_n coming after $n-1$ words $w_{1:n-1}$.

Occurrence near sentence terminals We want to use the notation $w_n|w_{1:n-1}$, with n fixed, for considering the event where w_n appears after the string $w_{k+1:n-1}$ appearing at the beginning of a sentence - distinct from the case where $w_{1:k}$ is some specific string. We accomplish this by setting $w_{1:k} = @^k$, where @ represents a special 'sentence terminal' word. This will allow us to write $Pr(w_n|w_{1:n-1})$ without being wrong.
Similarly, if $w_n = @$, $Pr(w_n|w_{1:n-1})$ denotes the probability of $w_{1:n-1}$ appearing at the end of a sentence.
Note that $Pr(w_1|@^m)$ actually represents the probability of w_1 appearing first in a sentence, and $Pr(w_2|@^{m-1}w_1)$ is the probability of w_2 appearing 2nd in a sentence after w_1. They are distinct from probabilities of occurrence of w_1 or w_2 after w_1 irrespective of position in the sentence.

12.4.1.2 Actual probability

As a sort of necessary preprocessing, ensure that w_m, the last word in the string is @, and $w_{1:n-1} = @^{n-1}$.
Then, the probability of generating a given m word string is exactly $Pr(w_{1:m}) = \prod_{k=n:m} Pr(w_k|w_{1:k-1})$.

12.4.1.3 Markov assumption

If one makes a simplifying nth order Markov assumption, which says that any word depends only on the previous $k <= n-1$ words, we get the approximation: $Pr(w_{1:m}) \approx \prod_{k=n:m} Pr(w_k|w_{k-n+1:k-1})$.

12.4.2 Estimation

$Pr(w_n|w_{1:n-1})$ are estimated by counting the number of occurrences of strings $w_{1:n}$ and $w_{1:n-1}$.

12.4.2.1 n and corpus size

Even in a large sized corpus, for large n, n-string sequences $w_{max(k-m+1,1):k}$ may be very rare; so it becomes difficult to estimate the necessary probabilities accurately.
One way of dealing with probabilities $Pr(w_n|w_{1:n-1})$ for which there is inadequate data is to replace them with $Pr(w_n|w_{2:n-1})$. In doing this, we have locally simplified the n-gram model into an $n-1$ gram model. One can even recursively reduce the model complexity until the data we have suffices to accurately estimate the simplified model. Thus, there is a tradeoff between accuracy/ complexity and estimability.
Also, storage space required to store model parameters grows exponentially with n.

12.4.2.2 Rare words

With rare words, one again encounters the problem of being able to estimate $Pr(w_n|w_{1:n-1})$ accurately with limited data. To deal with this, one often replaces rare word occurances with a special word UNK during preprocessing.

12.4.3 Smoothing

[**Incomplete**]

12.5 Partially observed states

12.5.1 Observations, states

(X_i) are called features/ covariates/ predictor/ input/ observed variables. (L_i) is the unobserved response/ state variable sequence. X_i, being a partially dependent of L_i, can be viewed as a partial observation of the state L_i.
The state space is $ran(L_i)$, while the observation space is $ran(X_i)$.

12.5.1.1 Use

These models are not only used for deriving models for f_X, but also for determining the state sequence L given X.

12.5.1.2 Applications

Spelling corrector, where X stands for the observed typed word and L stands for unobserved dictionary word.
Predicting part of speech is a classic application of HMM's.

12.5.2 Model classes

12.5.2.1 Generative model of Pr(X, L)

As in the case of general models of response variables, one may use these models to derive models for $f_{L|X}$ if needed.
This class of models includes HMM's.

12.5.2.2 Model L given X

Aka Conditional random field (CRF). Here, one uses a discriminative model $f_{L|X}$; so no effort is wasted in modeling f_X. The most common CRF is just a chain among L_i.

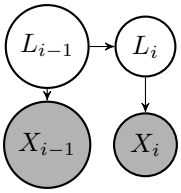
Ignoring sequentiality One can model $f_{L|X}(l|x) = \prod_i f_{L'|X'}(l_i|x_i)$. This model works surprisingly very well: Eg: In part of speech tagging, it yields around .95 correctness, while HMM may yield perhaps .02 more accuracy.
Note that this is not entirely same as Naive Bayes because $ran(X_i)$ may be multidimensional, and the model $f_{L'|X'}$ need not assume that these features are independent. So, any of the wide variety of classifiers may be used.

12.5.3 Partially observed state chain

Aka Bigram Hidden Markov Model (HMM).

12.5.3.1 Graphical model

The graphical model of the observation and label sequences has the following structure for $i = 2 : N$:



12.5.3.2 Representations

Parameters of a bigram HMM are the state transition probabilities $f_{L_t|L_{t-1}}$ and observation generation probabilities: $f_{X_t|L_t}$. As in the case of fully observed state chains, the state transition probabilities can be represented using a transition matrix or as labels of edges in a state transition graph which could now be expanded to include vertices corresponding to various observations.

12.5.3.3 Decoding/ filtering

Problem We want to find the most likely state sequence $X_1 : X_N$.

Message passing algorithm Viewing the state-chain as an equivalent undirected tree-structured graphical model, we can solve the problem using the divide and conquer max-product algorithm. When the messages passed during this computation are done in an order similar to that described in case of the forward-backward algorithm for finding marginal state distributions, we have the Viterbi algorithm.

12.5.3.4 Online label distribution inference

Problem Unlike the decoding problem, one is not satisfied with finding the most likely state sequence, the task is to find $f_{L_N|X_{1:N}=x_{1:N}}$ at time N.

Forward algorithm One can use a node elimination/ message passing algorithm applied to find the marginal probability distribution in the equivalent undirected tree structured graphical model. The node elimination ordering is: $1..N$. So, this is aka 'forward' algorithm.
This algorithm can be described inductively. At step t, suppose that one has determined $f_{L_{t-1},x_{1:t-1}}$, one simply does:

$$f_{L_t,x_{1:t}}(l_t) = \sum_{l_{t-1}} f_{L_{t-1},x_{1:t-1}}(l_{t-1}) f_{L_t|L_{t-1}}(l_t|l_{t-1}) f_{X_t|L=l_t}(x_t)$$

.
The base case is when $t = 1$, and $f_{L_1|x_1} = f_{X|L_1}(x_1) f_{L_1=l}$ can be easily determined. So, by induction it follows that we are able to determine $f_{L_N,X_{1:N}=x_{1:N}} \propto f_{L_N|X_{1:N}=x_{1:N}}$.

Analysis This is an $O(|T|^2)$ operation at each time step, where T is the state space.
If there are N time steps, the algorithm yields the correct result for $f_{L_N|x_{1:N}}$. But for $k < N$, $f_{L_k|x_{1:k}} \neq f_{L_k|x}$. This can be remedied by using the forward backward algorithm described below.

12.5.3.5 Past Label Distribution inference

Problem Aka Smoothing. One wants to find the sequence of distributions (f_{L_k}). This is a particular type of inference.

Algorithm One simply uses the node-elimination/ message passing sum-product dynamic programming algorithm to find marginal probabilities f_{L_k} of tree structured graphical models.
This is aka the forward backward algorithm when done for all k, and when computation is done in the following order: upchain 'messages' ($m_{t-1\to t}$) are passed first (in the forward step) and then down-chain 'messages' ($m_{t+1\to t}$) are passed (in the backward step).

Analysis The first step corresponds to the forward step described earlier, so $f_{L_i,x_{1:i}}$ is computed, at each node $i \in 1:N$.
In the backward step, at each node $i \in N:1$, $f_{x_{i+1:N}|L_i}$ is calculated. This can be described iteratively: suppose that $f_{x_{i+1:N}|L_i}$ is available, then one can find: $f_{x_{i:N}|L_{i-1}}(l_{i-1}) = \sum_{l_i} f_{x_{i+1:N}|L_i}(l_i) f_{L_i|L_{i-1}}(l_i|l_{i-1})$. So, it is in a way symmetric to the forward step.
Then, $f_{L_i,x} \propto f_{L_i|x}$ can be found by multiplying these.

12.5.3.6 Learning given (X, L) examples

Aka Supervised HMM learning. The basic idea is to use the empirical transition/ observation generation probabilities in order to estimate the HMM parameters.

Smoothing Especially in the case of observation generation probabilities, some smoothing is required: otherwise the distribution model would assign a probability of 0 to every state sequence l which might generate the observation sequence x containing a word not seen among the labeled examples. Also, it may be desirable to ensure that the observation generation probability $f_{X'|L'}(x'|l') > 0$ for any pair (x', l').

Reestimation using observation sequences Suppose that one also has access to samples of X, one can improve the parameter estimates by applying the EM algorithm described elsewhere for the case where only samples of X are provided.

12.5.3.7 Learning given observation samples X only

To do this, starting with an initial guess about the parameters $\theta^{(0)}$, one can iteratively produce parameters $\theta^{(i)}$ with greater likelihood values $f_{X|\theta}(x)$ from $\theta^{(i-1)}$ using the EM local optimization algorithm.
Two steps of the algorithm: 1] For each sample x: using $\theta^{(i-1)}$ with the forward-backward algorithm, infer distributions $f_{L_i|x}$; 2] Use this distribution to update maximum likelihood estimates [ie expectations] of the number of occurrences of $k \in ran(L_i)$, $(k_1, k_2) \in ran(L_i)^2$, (k, x); using which $\theta^{(i)}$ is computed as in the supervised case (: empirical emission and transition probabilities, possibly smoothed).

12.5.4 k-gram HMM

One can extend the notion of bigram HMM's to allow the current state to depend on previous $k-1$ states. Analogous the case of k-gram Markov chains, these can be reduced to bigram HMM's by expanding the state space to $ran(L_i)^{k-1}$. Thus, inference and learning algorithms for bigram HMM's can be adapted to work on k-gram HMM's.

12.6 Decision process

Stochastic processes where state sequence partially depends on a sequence of actions taken by an agent are described in the Machine intelligence survey.

13 Continuous response variables' prediction

Aka regression.
For overview, see Statistics survey. Here one models a (set of) response random variable Y in terms of input variables X.

13.1 Data preparation and assumptions

Saling, centering, addition of bias variables is assumed below. That, along with motivation, is described in the statistics survey.

13.2 Generalized linear model

13.2.1 Linear models

Here, we suppose that $L|X \sim XW + N$, where N is a 0-mean noise RV. Then, $E[L] = XW$, which is linear in parameters w.
Correpsonding to the constant variable $X_0 = 1$, we have bias parameters $W_{0,:}$.

13.2.2 Generalization

One can extend the family of linear models so that $E_{L|X}[L] = g^{-1}(XW)$ and $var[L] = f(E_{L|X}[L])$. Note that the variance is then a function of the predicted value.
A distribution from the exponential family must be used.

13.2.2.1 Log linear model

Aka poisson regression. $\log(E[L]) = XW$.

13.2.2.2 Logistic model

Aka logit model, logistic regression. A generalized linear model. See 'discriminative models of response' section.

13.2.2.3 Perceptron: step function

Here $E_{L|X}[X] = I[XW > 0]$.

13.3 Multi-layer generalized linear model

Aka Artificial Neural Network, multi-layer perceptron (a misnomer given that the activation function described below is not the non-differentiable step function).

13.3.1 Model

Suppose one wants to predict $Y = y$ using the input $X^{(0)} = x^{(0)}$ (aka input layer). The model $Y = h(X^{(0)})$ is hierarchical.
One can obtain layer upon layer of intermediary random variables $X^{(j)} = \left\{X_i^{(j)}\right\}$, where $X_i^{(j)} = f(\langle w_i^{(j)}, X_i^{(j-1)}\rangle + w_{i,0}^{(j)})$. Suppose one has k such intermediary layers. One finally models $X_j^{(k+1)} = h(\langle w_j^{(k+1)}, X^{(k)}\rangle)$ (aka the output layer).

13.3.1.1 Component names

The intermediary layers are called hidden layers. Neurons in the hidden/ 'skip' layers are called hidden units. Neurons in the output layer are called output units.
$a_i^{(j)} = \langle w_i^{(j)}, X_i^{j-1}\rangle + w_{i,0}^{(j)}$ is called the activation.

13.3.1.2 Activation function

f is usually a non-linear function - the logistic step function with the range [-1, 1] and the tanh function are commonly used in case of classification problems being solved by relaxation to regression problem. In case of regression problems or in case of 'skip' layer variables, the final f is just the identity function - or a sigmoid function which approximates it.

13.3.1.3 Visualization as a network

There is the input layer, hidden layers and the output layer. Directed arrows go from one layer to the next. This is a Directed Graphical Model except that the intermediary dependencies are deterministic, not stochastic.

13.3.1.4 Nomenclature

Depending on preference, a model with K layers of non-input (intermediary + output) variables is called a $K + 1$ or K layer neural network. We prefer the latter.
2 layer networks are most common.

13.3.2 Connection to other models

[**Incomplete**]

13.3.3 Model training

One can write $Y = h(X)$ where h is a differentiable, yet non-convex function. One can fit model parameters to training data $((x_i, l_i))$ by minimizing (possibly regularized) empirical loss.

13.3.3.1 Gradient finding

Given an error fn $E(y)$ for a given data point (x, t), various optimization techniques require one to find $\nabla_w E(y)$. This gradient can be found efficiently using the error back-propagation algorithm.
The idea is that the parameter $w_{k,j}^{(f)}$ only affects $E(y)$ through the output : $X_k^{(f)}$, so one can apply the chain rule for partial derivatives.
For output unit, $\frac{dE(X_1^{(t)})}{dw_{1,j}^{(t)}} = \frac{dE(X_1^{(t)})}{dX_1^{(t)}} f'(a_1^{(t)}) X_j^{(t-1)}$. Denote $d_1^t := \frac{dE(X_1^{(t)})}{dX_1^{(t)}} f'(a_1^{(t)})$ - the quantity multiplied with $X_j^{(t-1)}$ in the expression. This is aka 'error'.
Assume that $\frac{dE(X_1^{(t)})}{dw_{i,j}^{(f)}} = d_i^{(f)} X_j^{(f-1)}$ holds for neurons in the levels $f : t$. We can see that a similar expression holds for level $f - 1$ too.
For symbol manipulation convenience, set ith input to kth neuron in layer f: $Z_{k,i}^f = X_i^{f-1}$. Using chain rule for partial derivatives:
$\frac{dE(X_1^{(t)})}{dw_{i,j}^{(f-1)}} = \sum_k \frac{dE(X_1^{(t)})}{dZ_{k,i}^f} \frac{dX_i^{(f-1)}}{dw_{i,j}^{(f-1)}} = \sum_k d_k^f f'(a_i^{(f-1)}) X_j^{(f-2)}$.
Setting $d_i^{(f-1)} = \sum_k d_k^f f'(a_i^{(f-1)})$, we see from mathematical induction that $\frac{dE(X_1^{(t)})}{dw_{i,j}^{(f-1)}}$ can be calculated for all neurons given the 'error' for the layer ahead.
So, the back propagation algorithm to find the gradient is: First run the neural network with input x and record all outputs X_j^k. Starting with the output layer, determine the error $d_i^{(f-1)}$ and thence the appropriate gradient components.

13.3.3.2 Weight initialization

Starting point for (stochastic) gradient descent is done as follows. Weights can be initialized randomly with mean 0 and standard deviation $1/m^2$, where m is the fan-in of a unit.

13.3.4 Flexibility

There are theorems which show that a two layer network can approximate any continuous function to arbitrary accuracy - provided a sufficient number of intermediary variables are allowed!
The flexibility of the multi-layer generalized linear model derives from the non-linearity in the activation functions.

13.3.5 Disadvantages

Objective function minimized during training is non-convex.

Large diversity in training examples required. The model learned is not accessible for use in modeling the process producing the data realistically, though it may be effective.
It can be inefficient in terms of storage space and computational resources required.
The brain by contrast solves all these problems because: its hardware is tuned to the neural network architecture; its training examples have suffiencient variety.

13.4 Deep belief network

Extending the idea of neural networks, adding structure to it and using a sort of L1 regularization to make the network sparse, one gets deep belief networks. These have proved to be very successful in many applications since 2007.
[**Incomplete**]

Part IV

References

Bibliography

[1] *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.