Machine Learning 2 - Cheat Sheet
Andrea Jemmett - May 23, 2016

1 Probability Theory

1.1 Independence

$p(X,Y) = p(X)p(Y) \Leftrightarrow p(X|Y) = p(X) \Leftrightarrow p(Y|X) = p(Y)$

1.2 Conditional Independence

$X \perp\!\!\!\perp Y \mid Z \Longleftrightarrow p(X,Y|Z) = p(X|Z)p(Y|Z)$
$p(X,Y|Z) = p(X|Y,Z)p(Y|Z) = p(X|Z)p(Y|Z)$

1.3 Sum and Product Rules

$p(X,Y) = p(X)p(Y|X), \qquad p(X,Y,Z) = p(X)p(Y|X)p(Z|X,Y)$
$p(X) = \sum_Y p(X,Y)$

1.4 Bayes' Theorem

$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}, \qquad p(Y|X,Z) = \frac{p(X|Y,Z)p(Y|Z)}{p(X|Z)}$

1.5 Chain Rule

$p(x_a,x_b,x_c) = p(x_c|x_a,x_b)p(x_b|x_a)p(x_a) \qquad p(\mathbf{x}) = \prod_{k=1}^K p(x_k|\mathrm{pa}_k)$

1.6 Misc

$p(a,b|c) = \frac{p(a,b,c)}{p(c)} \qquad p(a,b,c) = p(a|c)p(b|c)p(c)$

2 Distributions

Binary	Bernoulli	Binomial	Beta
Discrete	Categorical	Multinomial	Dirichlet

2.1 Bernoulli Distribution

$\mathrm{Ber}(x|n) = \mu^x(1-\mu)^{1-x}, \qquad \mathbb{E}[x] = \mu, \qquad Var[x] = \mu - \mu^2,$
$P(D,\mu) = \prod_{n=1}^N \mu^{x_n}(1-\mu)^{1-x_n}, \qquad \mu_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^N x_n$

2.2 Binomial Distribution

$\mathrm{Bin}(m|N,\mu) = \binom{N}{m}\mu^m(1-\mu)^{N-m}, \qquad \frac{n!}{k!(n-k)!} = \binom{n}{k},$
$\mathbb{E}[m] = N\mu, \qquad Var[m] = N\mu(1-\mu), \qquad \mu_{\mathrm{ML}} = \frac{m}{N}$

2.3 Categorical Distribution

$p(\mathbf{x}|\boldsymbol{\mu}) = \prod_k \mu_k^{x_k}, \qquad \boldsymbol{\mu} \in \{0,1\}^K, \qquad \sum_k \mu_k = 1, \qquad \boldsymbol{\mu}_{\mathrm{ML}} = \frac{\mathbf{m}}{N},$
$m_k = \sum_n x_{nk}, \qquad \mathrm{Mult}(m_1 \dots, m_k|N,\boldsymbol{\mu}) = (\frac{N!}{m_1!,\dots,m_k!}\prod_k)\mu_k^{mk}$

2.4 Beta Distribution

$\mathrm{Beta}(\mu|a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\mu^{a-1}(1-\mu)^{b-1}$, [1] $\qquad \mathbb{E}[\mu] = \frac{a}{a+b},$
$Var[x] = \frac{ab}{(a+b)^2(a+b+1)}, \qquad p(\mu|m,l,a,b) \propto \mu^{m+a-1}(1-\mu)^{l+b-1}$

2.5 Gamma Distribution

$\mathrm{Gamma}(\tau|a,b) = \frac{b^a}{\Gamma(a)}\tau^{a-1}e^{-b\tau}, \qquad \mathbb{E}[\tau] = \frac{a}{b}, \qquad Var[\tau] = \frac{a}{b^2},$
$\mathrm{mode}[\tau] = \frac{a-1}{b} \text{ for } a \geq 1, \qquad \mathbb{E}[\ln\tau] = \psi(a) - \ln b,$
$H(\tau) = ln\Gamma(a) - (a-1)\psi(a) - \ln b + a$

2.6 Multinomial Distribution

$\mathbf{x} = [0,0,0,0,1,0,0]^\mathsf{T}, \qquad \sum_{k=1}^K x_k = 1, \qquad p(\mathbf{x}|\boldsymbol{\mu}) = \prod_{k=1}^K \mu_k^{x_k},$
$\sum_{k=1}^K \mu_k = 1, \qquad \mu_k^{\mathrm{ML}} = \frac{m_k}{N}, \qquad m_k = \sum_{k=1}^K x_{nk}$

2.7 Dirichlet Distribution

$\mathrm{Dir}(\boldsymbol{\mu}|\boldsymbol{\alpha}) = \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1)\cdots\Gamma(\alpha_k)}\prod_{k=1}^K \mu_k^{\alpha_k - 1}, \qquad \alpha_0 = \sum_{k=1}^K \alpha_k$

2.8 Gaussian Distribution

$\mathcal{N}(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{1}{2\sigma^2}(x-\mu)^2),$
$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = (2\pi)^{-\frac{D}{2}}|\boldsymbol{\Sigma}|^{-\frac{1}{2}}\exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$

[1] $\Gamma(x) = \int_0^1 u^{x-1}e^{-u} = 1, \qquad \Gamma(x+1) = \Gamma(x)x, \qquad \Gamma(x+1) = x!$
[2] $B(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$

2.8.1 ML for the Gaussian

$\ln p(X|\boldsymbol{\mu},\boldsymbol{\Sigma}) = -\frac{ND}{2}\ln(2\pi) - \frac{N}{2}\ln|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{n=1}^N(\mathbf{x}_n-\boldsymbol{\mu})^\mathsf{T}\boldsymbol{\Sigma}^{-1}(\mathbf{x}_n-\boldsymbol{\mu}),$
$\boldsymbol{\mu}_{\mathrm{ML}} = 1/N\sum_{n=1}^N \mathbf{x}_n, \qquad \boldsymbol{\Sigma}_{\mathrm{ML}} = 1/N\sum_{n=1}^N(\mathbf{x}_n-\boldsymbol{\mu})^\mathsf{T}(\mathbf{x}_n-\boldsymbol{\mu})$

2.8.2 Stochastic gradient descent Gaussian

$\max\ P(x_1,\cdots,x_n|\theta),\ \theta^N = \theta^{N-1} + \alpha_{N-1}\frac{\partial}{\partial\theta^{N-1}}\ln\ p(x_n|\theta^{N-1})$

2.8.3 Marginal and Conditional Gaussians

Given $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Lambda}^{-1})$ and $p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\mathbf{x}+\mathbf{b},\mathbf{L}^{-1})$. We get
$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu}+\mathbf{b},\mathbf{L}^{-1}+\mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^\mathsf{T})$ and
$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\Sigma}[\mathbf{A}^\mathsf{T}\mathbf{L}(\mathbf{y}-\mathbf{b})+\boldsymbol{\Lambda}\boldsymbol{\mu}],\boldsymbol{\Sigma})$
where $\boldsymbol{\Sigma} = (\boldsymbol{\Lambda}+\mathbf{A}^\mathsf{T}\mathbf{L}\mathbf{A})^{-1}$.

2.9 Student's T distribution

The heavy tail of the student-t distribution makes it more robust against outliers.
$St(x|\mu,\lambda,\nu) = \frac{\Gamma(\nu/2+1/2)}{\Gamma(\nu/2)}(\frac{\lambda^{1/2}}{(\pi\nu)^{D/2}})[1+\frac{\lambda(x-\mu)^2}{\nu}]^{-\nu/2-D/2},$
$f_x(x) = \frac{\Gamma[(\nu+p)/2]}{\Gamma(\nu/2)\nu^{p/2}\pi^{p/2}|\Sigma|^{1/2}[1+1/\nu(x-\mu)^T\Sigma^{-1}(x-\mu)]^{(\nu+p)/2}}$
$\mathbb{E}(\mathbf{x}) = \frac{\Gamma(D/2+v/2)}{\Gamma(v/2)}\frac{|\Lambda|^{1/2}}{(\pi v)^{D/2}}\int[1+\frac{(x-\mu)^T\Lambda(x-\mu)}{v}]^{-D/2-v/2}x\mathrm{d}x$

3 Independent Component Analysis

We have a set of N observations:
$D = \{\mathbf{x}^{(n)} \in \mathbb{R}^J\}_{n=1}^N$
, with I independent sources. Thus:
$\mathbf{x} = \mathbf{G}\mathbf{s}$, where G is not known. We assume that the latent variables are independently distributed, with marginal distributions $P(s_i|H) = p_i(s_i)$. The probability of the observables and the hidden nodes is:
$P(\{x^{(n)},s^{(n)}\}_{n=1}^N|G,H) = \prod_{n=1}^N[P(x^{(n)}|s^{(n)},G,H)P(s^{(n)}|H)]$
$= \prod_{n=1}^N[(\prod_J \delta(x_j^{(n)} - \sum_i G_{ji}s_i^{(n)}))(\prod_i p_i(s_i^{(n)}))].$
The factor x is generated without noise. For learning the G from the data the likelihood is: $P(D|G,H) = \prod_n P(x^{(n)}|G,H)$
, which is a product of factors if we marginalize over the latent variables. We can express each coefficient of $\mathbf{G}$, as a summation over all coefficients multiplied by a delta function such that, $G_{ji}s_i^{(n)} = \sum_i G_{ji}s_i^{(n)}$. ($\mathcal{H}$ denotes the model.)
$p(\mathbf{x^{(n)}}|\mathbf{G},\mathcal{H}) = \int p(\mathbf{x^{(n)}}|\mathbf{s}^{(n)},\mathbf{G},\mathcal{H})p(\mathbf{s}^{(n)}|\mathcal{H})\mathrm{d}^I\mathbf{s}^{(n)}$
$= \int\prod_j \delta(x_j^{(n)} - G_{ji}s_i^{(n)})\prod_i p_i(s_i^{(n)})\mathrm{d}^I\mathbf{s}^{(n)}$
$= \frac{1}{|\det\ \mathbf{G}|}\prod_i p_i(G_{ji}^{-1}x_j) \Rightarrow$
$\ln p(\mathbf{x^{(n)}}|\mathbf{G},H) = -\ln|\det\mathbf{G}| + \sum_i \ln p_i(G_{ji}^{-1}x_j)$
, which is the log-likelihood of the data D. Now, to find the gradient of the log-likelihood, we are introducing $\mathbf{W} = \mathbf{G}^{-1}$, now the log-likelihood can be written as:
$\ln p(\mathbf{x^{(n)}}|\mathbf{G},H) = -\ln|\det\mathbf{W}| + \sum_i \ln p_i(W_{ji}x_j)$
The gradient of the log-likelihood equation will be:
$\frac{\partial}{\partial W_{ij}}\ln p(\mathbf{x^{(n)}}|\mathbf{G},\mathcal{H}) = -\frac{\partial}{\partial W_{ij}}(\ln|\det\mathbf{W}|) + \frac{\partial}{\partial W_{ij}}(\sum_i \ln p_i(W_{ji}x_j))$

4 Generative Models for Discrete Data

We can classify a feature vector $\mathbf{x}$ using the Bayes rule
$p(y=c|\mathbf{x},\boldsymbol{\theta}) \propto p(\mathbf{x}|y=c,\boldsymbol{\theta})p(y=c|\boldsymbol{\theta})$
We can use different models for the data when it's discrete, based on what kind of distribution we expect the data to assume and a respective *conjugate* prior over the model parameters $\boldsymbol{\theta}$.

4.1 Beta-Binomial Model

In this model we can observe a series of Bernoulli trials (e.g. coin tosses) or the number of heads (and the number of tails or total number of tosses), which is a Binomial, and it would result in the same **likelihood**: $p(\mathcal{D}|\theta) = \theta^{N_1}(1-\theta)^{N_0}$. A conjugate **prior** for this likelihood is given by $\mathrm{Beta}(\theta|a,b) \propto \theta^{a-1}(1-\theta)^{b-1}$. The **posterior** is then obtained by multiplying the prior with the likelihood, $p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta) = \mathrm{Bin}(N_1|\theta,N_1+N_0)\mathrm{Beta}(\theta|a,b) \propto \mathrm{Beta}(\theta|a+N_1,b+N_0)$. The **evidence** is obtained from $p(\theta|\mathcal{D}) = \frac{1}{p(\mathcal{D})}p(\mathcal{D}|\theta)p(\theta)$ normalization of posterior is $1/B(a+N_1,b+N_0)$[2], of prior is $1/B(a,b)$, hence $p(\mathcal{D}) = \binom{N}{N_1}B(a+N_1,b+N_0)/B(a,b)$.

4.2 Dirichlet-Multinomial Model

For example N dice rolls or *multinomial* events (with K outcomes). The **likelihood** is $p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{k=1}^K \theta_k^{N_k}$ where N_k counts times event k occurred (it's a suff. statistic). A conjugate **prior** is the Dirichlet distribution. The **posterior** is $p(\boldsymbol{\theta}|\mathcal{D}) \propto \mathrm{Dir}(\boldsymbol{\theta}|\boldsymbol{\alpha})p(\mathcal{D}|\boldsymbol{\theta}) \propto \mathrm{Dir}(\boldsymbol{\theta}|\alpha_1+N_1,\dots,\alpha_k+N_k)$. The **evidence** is (obtained as the previous model) $p(\mathcal{D}) = B(\mathbf{N}+\boldsymbol{\alpha})/B(\boldsymbol{\alpha})$.

5 Graphical Models

Capture noise, for reasoning, enormous datasets, for causality, for designing models, CIR are encoded in the graph, for inference.

5.1 Discriminative

$p(\mathbf{t}|w) = p(w)\prod_{n=1}^N p(t_n|w)$

5.2 Generative

$T^* = P(T^*|W,X^*,X_i,T_i,\sigma^2,a) =, [\prod_{i=1}^N P(X_i|W,\sigma^2)]P(W|a)P(T^*|X^*,W,\sigma^2)$

5.3 Directed GMs a.k.a. Bayesian Networks

Nodes are connected by *directed* arrows. The full joint distribution is $p(\mathbf{x}) = \prod_{k=1}^K p(x_k|\mathrm{pa}(x_k))$
Directed Acyclic Graphs are BNs without *directed* loops.

5.3.1 Blocking Rules

○ → ● → ○ ○ ← ● → ○ ○ → ⋯ ○ ⋯ ← ○

5.3.2 D-separation

$A \perp\!\!\!\perp B \mid C$ holds if each path that connects a node in A with a node in B is *blocked*, that is
a) the arrows on the path meet either head-to-tail or tail-to-tail at the node and the node is in C, *or*
b) the arrows meed head-to-head at the node and either the node nor any of its descendants is in C.

5.4 Markov Random Fields

Graphical Models with undirected edges (a.k.a. Undirected Graphical Models).

5.4.1 Conditional Independence

$A \perp\!\!\!\perp B \mid C$ holds if all paths connecting every node in A to every other node in B is 'blocked' by a node in C (blocked means passing through).

5.4.2 Cliques and Maximal Cliques

A *clique* is a subset of nodes in the graph such that there exists a link between all pairs of nodes in the subset (the set of nodes in a clique is fully connected). A *maximal clique* is a clique such that it is not possible to include any other nodes from the graph in without it ceasing to be a clique.

5.4.3 Factorization

A MRF can be factorized using *potential functions* over its maximal cliques: $p(\mathbf{x}) = \frac{1}{Z}\prod_C \psi_C(\mathbf{x}_C) \quad Z = \sum_x \prod_C \psi_C(\mathbf{x}_C)$

5.4.4 Relation to Directed Graphs

To transform a directed graph into an undirected one we have to perform a *moralization* process of "marrying the parents" of each node (by linking them) and removing all remaining arrows.

5.5 Markov Blanket

$p(x_i|x_{\mathrm{MB}_i}, x_{\mathrm{rest}}) = p(x_i|x_{\mathrm{MB}_i})$
Directed case: The MB of x_i consists of: the parents of x_i, the children of x_i and the co-parents of the children of x_i.
Undirected case: All neighboring nodes of x_i.

5.6 Naive Bayes

The problem of finding a label y^* for some previously unobserved vector of features $\mathbf{x}^*$, while having previously observed $\{\mathbf{x}_n, y_n\}_{n=1,\dots,N}$. We build a *generative* GM, linking the label y_n with each feature of the feature vector $\mathbf{x}_n$. This implies that all features are independent of each other given the label. For a single data case the joint probability is
$p(y, x_1, \dots, x_D|\boldsymbol{\eta}, \boldsymbol{\theta}_i) = p(y|\boldsymbol{\eta}) \prod_{i=1}^D p(x_i|\boldsymbol{\theta}_i)$
while the probability of the full dataset is $\prod_{n=1}^N p(y_n|\boldsymbol{\eta}) \prod_{i=1}^D p(x_i|\boldsymbol{\theta}_i)$
Note that this generative form is chosen as the other option (features pointing towards the label) would imply a very highly parameterized model, as we would considering $p(y|x_1, \dots, x_D)$.
Finding a label y^* implies finding
$y^* = \mathrm{argmax}_y \left\{ \ln p(y|\boldsymbol{\eta}) + \sum_{i=1}^D \ln p(x_i^*|y, \boldsymbol{\theta}_i) \right\}$

5.7 Maximum Likelihood Training in BNs

It is fast because the log-likelihood *decomposes* into a sum over all variables X_i. Learning all parameters reduces into a collection of independent tasks of learning $p(x_i|\mathrm{pa}_{x_i})$.
$p(x_i|\mathrm{pa}_{x_i}) = \frac{N(x_i, \mathrm{pa}_{x_i})}{N(\mathrm{pa}_{x_i})}$
is the number of times x_i co-occurred with pa_{x_i} divided by the number of times pa_{x_i} occurred.
The probability of a set of variables $\mathbf{x}$ in a Bayes Net is defined as $p(x) = \prod_i p(x_i|x_{pa_i})$. If we define a function $\theta(x_i, x_{pa_i})$ and use it instead of $p(x_i|x_{pa_i})$, we have $p(x) = \prod_i \theta(x_i, x_{pa_i})$, if we constrain θ such that $\sum_{x_i} \theta(x_i, x_{pa_i}) = 1, \forall i$. We now have that the probability of the full data set: $p(\{\widetilde{x}_{in}\}) = \prod_n \prod_i \prod_{x_i} \prod_{x_{pa_i}} \theta(x_i, x_{pa_i})^{\mathbb{I}[x_i = \widetilde{x}_{in} \wedge x_{pa_i} = \widetilde{x}_{pa_i n}]}$

5.8 Inference in Graphical Models

In which some of the variables are observed and we wish to compute the posterior distribution of one or more subsets of other variables.

5.8.1 Inference on a chain

$p(\mathbf{x}) = p(x_1, \dots, x_N)$
$= \frac{1}{Z}\psi_{1,2}(x_1, x_2) \dots \psi_{N-1,N}(x_{N-1}, x_N)$
$= \sum_{x_1} \sum_{x_2} \cdots \sum_{x_{N-1}} \sum_{x_N} p(\mathbf{x})$
$= \sum_{x_1} \cdots \sum_{x_{n-1}} \sum_{x_{n+1}} \cdots \sum_{x_{N-1}} \sum_{x_N} p(\mathbf{x})$
$= \frac{1}{Z} \sum_{x_1} \cdots \sum_{x_{n-1}} \psi_{x_1,x_2} \dots \psi_{x_{n-1},x_n} \mu_\beta(x_n)$
$\mu_\beta(x_n) = \frac{1}{Z} \sum_{x_{n+1}} \psi_{x_n,x_{n+1}} \cdots \sum_{x_N} \psi_{x_{N-1},x_N}$
$\mu_\alpha(x_n) = \frac{1}{Z} \sum_{x_{n-1}} \psi_{x_{n-1},x_n} \cdots \sum_{x_1} \psi_{x_2,x_1}$
$p(x_n) = \frac{1}{z}\mu_\alpha(x_n)\mu_\beta(x_n) \qquad O(NK^2)$

5.8.2 Factor Graphs

A tree is a graph with no loops. Both directed and undirected trees can be converted to a factor graph tree, but a directed tree could result in a non-tree structure when converted to an undirected representation. It is called a poly-tree (and not simply a tree) since its undirected representation (middle graph) includes a loop. The factor graph representation is again a tree. Factor graphs are the most general representation, and since any other tree representation can be easily converted to a factor tree, the sum-product algorithm is defined for factor trees.

5.8.3 Sum-Product Algorithm

It is used to find *individual* marginals.
Probability of the factor graph: $p(\mathbf{x}) = \frac{1}{Z}\prod_\alpha f_\alpha(\mathbf{x}_\alpha)$
factor → variable message:
$\mu_{f_\alpha \to x_i}(x_i) = \sum_{\mathbf{x}_\alpha \setminus x_i} f_\alpha(\mathbf{x}_\alpha) \prod_{m \in \mathrm{ne}(f_\alpha) \setminus x_i} \mu_{x_m \to f_\alpha}(x_m)$
variable → factor message:
$\mu_{x_m \to f_\alpha}(x_m) = \prod_{\beta \in \mathrm{ne}(x_m) \setminus f_\alpha} \mu_{f_\beta \to x_m}(x_m)$

leaf nodes x_l is variable: $\mu_{x_l \to f_\delta}(x_l) = 1$
f_ε is factor: $\mu_{f_\varepsilon \to x_k}(x_k) = f_\varepsilon(x_k)$
variable marginal: $p(x) = \prod_{\alpha \in \mathrm{ne}(x)} \mu_{f_\alpha \to x}(x)$
factor marginal: $p(\mathbf{x}_\alpha) = f_\alpha(\mathbf{x}_\alpha) \prod_{i \in \mathrm{ne}(f_\alpha)} \mu_{x_i \to f_\alpha}(x_i)$

By caching intermediate results and calculating two messages for each edge in a tree-structured factor graph, the Sum-Product algorithm efficiently computes all variable marginals in $O(2EK^2)$ (where E is the number of edges and each variable is assumed to have K possible values).

5.8.4 Max-Sum Algorithm

It is used to find a setting of the variables that has the largest *joint* probability and the value of that probability.
$\mathbf{x}^{\max} = \operatorname{argmax}_{\mathbf{x}} \prod_\alpha f_\alpha(\mathbf{x}_\alpha) \qquad p(\mathbf{x}^{\max}) = \max_{\mathbf{x}} \frac{1}{Z} \prod_\alpha f_\alpha(\mathbf{x}_\alpha)$
$\log p(\mathbf{x}^{\max}) = \max_{\mathbf{x}} \left\{ \log Z + \sum_\alpha f_\alpha(\mathbf{x}_\alpha) \right\}$
factor → variable message:
$\mu_{f_\alpha \to x_i}(x_i) = \max_{\mathbf{x}_\alpha \setminus x_i} \left[(\ln f_\alpha(\mathbf{x}_\alpha)) + \sum_{m \in \mathrm{ne}(f_\alpha) \setminus x_i} \mu_{x_m \to f_\alpha}(x_m) \right]$
variable → factor message:
$\mu_{x_m \to f_\alpha}(x_m) = \sum_{f_\beta \in \mathrm{ne}(x_m) \setminus f_\alpha} \mu_{f_\beta \to x_m}(x_m)$

leaf nodes x_l is variable: $\mu_{x_l \to f_\delta} = 0$
f_ε is factor: $\mu_{f_\varepsilon \to x_k} = \ln f_\varepsilon(x_k)$
at root $p^{\max} = \max_x \left[\sum_{f_\alpha \in \mathrm{ne}(x)} \mu_{f_\alpha \to x}(x) \right]$
$x^{\max} = \operatorname{argmax}_x \left[\sum_{f_\alpha \in \mathrm{ne}(x)} \mu_{f_\alpha \to x}(x) \right]$
max-marginals $q(x_i) = -\ln Z + \sum_{f_\alpha \in \mathrm{ne}(x_i)} \mu_{f_\alpha \to x_i}(x_i)$
Given max-marginals you have to perform a decoding step (the Viterbi algorithm) in order to find the global optimum. If $q(x_i)$ has unique maximum, we can use $x_i^* = \operatorname{argmax}_{x_i} q(x_i)$.

5.8.5 Variable Elimination

If we have some clusters C_i and C_j, then:
$S_{ij} = C_i \cap C_j$ Define potential functions Ψ_i for each cluster C_i: $\forall C_i : \Psi_i \coloneqq \prod_{\psi_j \in C_i} \psi_j$ $m_i \to C_j = \sum_{C_i \setminus S_{ij}} \Psi_i \prod_{k \in \mathrm{ne}(i) \setminus j} m_{k \to i}(C_i)$
Note that: $C_i \setminus S_{ij} = C_i \setminus C_j$
Belief: $\Psi_{\mathrm{root}} \prod_{k \in \mathrm{ne}(\mathrm{root})} m_{k \to \mathrm{root}}(C_{\mathrm{root}}) \propto p(C_{\mathrm{root}})$
Clusters replace factors, edges replace variables.
Tree width The width of a tree is one smaller than the biggest clique in the optimal ordering. Finding the optimal order is NP-hard.
Running Intersection Property A clique tree has the running intersection property if $x \in C_i \wedge x \in C_j \Rightarrow x \in C_k$ on the clique path between C_i and C_j.

5.8.6 Loopy Belief Propagation

Is a way to approximate inference in graphs with loops. The idea is simply to apply the sum-product algorithm even though there is no guarantee that it will yield good results. This approach is possible because the message passing rules are purely local. However, because of the loops, information can flow many times around the graph; for some models it will converge, for others it will not. We will say that a node a has a **pending** message on its link to a node b if node a has received any message on any of its other links since the last time it sent a message to b. Thus, when a node receives a message on one of its links, this creates pending messages on all of its other links. Then we need to send only pending messages otherwise they will be duplicate. When there are no more pending messages, the product of the received messages at every node gives the (exact for tree) marginal. For loopy graphs however the algorithm may not converge and when we stop the marginals are *approximate*.

6 EM, VM, VB

EM, VEM are learning algorithms, VB is for inference.
If $Q = P$, use EM. If Q is simpler than P, use VEM (easier computation, might never reach max, biased result)
Bayes: approximate $P(\theta|X)$ (eg sampling). Maximize both θ and α.

Frequentist:	EM: unbiased, can overfit	VEM: biased, can overfit
Bayesian:	$P(\theta\|X)$: best, but comp.hard, no overfitting bc no fitting variance	VB

Bayes does not overfit because you integrate over parameter space.

6.1 EM for Gaussian Mixtures

1. Initialize parameters $\theta = (\mu_{1:k}, \pi_{1:k}, \Sigma_{1:k})$
2. **E-step**: evaluate responsibilities using current params
   $\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(x_n|\mu_k,\Sigma_k)}{\sum_k \pi_k \mathcal{N}(x_n|\mu_k,\Sigma_k)}$
3. **M-step**: re-estimate params using current responsabilities
   $\mu_k^{\text{new}} = \frac{1}{N_k}\sum_{n=1}^N \gamma(z_{nk}) x_n$
   $\Sigma_k^{\text{new}} = \frac{1}{N_k}\sum_{n=1}^N \gamma(z_{nk})(x_n - \mu_k^{\text{new}})(x_n - \mu_k^{\text{new}})^\top$
   $\pi_k^{\text{new}} = \frac{N_k}{N}$
   where $N_k = \sum_{n=1}^N \gamma(z_{nk})$
4. Evaluate the log-likelihood
   $\ln p(X|\mu,\Sigma,\pi) = \sum_{n=1}^N \ln\{\sum_{k=1}^K \pi_k \mathcal{N}(x_n|\mu_k,\Sigma_k)\}$
   and check for convergence of either the parameters or the log-likelihood.

6.2 The General EM Algorithm

1. Choose an initial setting for the parameters θ^{old}
2. **E-step**: evaluate $p(Z|X,\theta^{\text{old}})$
3. **M-step**: evaluate θ^{new}
   $\theta^{\text{new}} = \text{argmax}_\theta\, Q(\theta,\theta^{\text{old}})$
   where $Q(\theta,\theta^{\text{old}}) = \sum_Z p(Z|X,\theta^{\text{old}}) \ln p(X,Z|\theta)$
4. Check for convergence of either the log likelihood or the parameter values. If the convergence criterion is not satisfied, then let $\theta^{\text{old}} \leftarrow \theta^{\text{new}}$ and return to step 2.

If optimization of $p(X|\theta) = \sum_Z p(X,Z|\theta)$ is hard, but not that of $p(X,Z|\theta)$ directly. Then we define a distribution over latent variables $q(Z)$ so that we have $\ln p(X|\theta) = \mathcal{L}(q,\theta) + \text{KL}(q||p)$ where $\mathcal{L}(q,\theta) = \sum_Z q(Z) \ln \frac{p(X,Z|\theta)}{q(Z)}$.
In the E-step we maximize the lower bound $\mathcal{L}$ w.r.t. $q(Z)$ while holding θ^{old} fixed. In the M-step the lower bound is maximized w.r.t. θ while holding θ^{old} fixed to obtain θ^{new}.

6.3 Variational EM (learning)

Variational EM algorithm has two steps:
E-Step:
Given θ^t, evaluate $q_n^t(z_n|x_n) = p(z_n|x_n,\theta^t)$, (or increase $\mathcal{L}(\theta^t,q)$ over q). This is not easy to solve, but sometimes it can be done (for example in Mixture of Gaussians).
[With reference to lec. notes 07] The green dot upon the circle will be the best q that approximates p as it is the shortest point from p.
M-Step:
Given q^t, solve $\theta^{t+1} = \text{argmax}_{\theta^t} \sum_n \mathbb{E}_{q_n^t}[\ln p(x_n,z_n|\theta^t)]$, (or increase $\mathcal{L}(\theta,q^t)$ over θ). We maximize the lower bounds until we find the maximum. The convergence is rather slow. In terms of speed is not the best algorithm.
Proof of Convergence:
We have to prove that: $h(\theta_t) \leq \ldots \leq h(\theta_{t+1})$
$h(\theta_t) = \mathcal{L}(\theta_t, p(z|x,\theta_t))$ $h(\theta_t) \leq \mathcal{L}(\theta_{t+1}, p(z|x,\theta_t))$, (M-Step): We maximise over θ_{t+1}. So $\mathcal{L}$ in eq.(13) must be bigger than the one in the (12). In the next step we have: $h(\theta_t) = l(\theta_{t+1}) - \mathcal{KL}[q_t||p(z|x,\theta_{t+1})]$, $(l = \mathcal{L}+\mathcal{KL})$ The distribution in Kullback-Leibler divergence are not the same so $\mathcal{KL}$ is not equal to 0 at this occasion, due to the fact that the two terms of $\mathcal{KL}$ divergence are coming from different iterations. Hence, $h(\theta_t) \leq h(\theta_{t+1})$, because $\mathcal{KL} \geq 0$.
Lower bound $\mathcal{L}(q)$ and KL-divergence $\ln p(X) = \mathcal{L}(q) + \mathcal{KL}(q||p)$
$\mathcal{L}(q) = \int q(Z) \ln\{\frac{p(X,Z)}{q(Z)}\} dZ$ $\quad$ $\mathcal{KL}(q||p) = -\int q(Z) \ln\{\frac{p(Z|X)}{q(Z)}\} dZ$
Note that
$Q(\theta,\theta^{old}) = \sum_z p(Z|X,\theta^{old}) \ln p(X,Z|\theta) = \mathbb{E}_{Z|X,\theta^{old}}[\ln p(X,Z|\theta)]$

6.4 Variational Bayes (inference)

Instead of treating θ as parameter, let us treat it as a hidden random variable and calculate its posterior. We assume a factorized distribution over the parameters so that $q(\theta) = \prod_{i=1}^D q(\theta_i)$.
$\mathcal{L}(q) = \int \prod_i q_i \{\ln p(X,\theta) - \sum_i \ln q_i\} \text{d}\theta = \int q_j \mathbb{E}_{i\neq j}[\ln p(X,\theta)] + \text{const}$
$\mathbb{E}_{i\neq j}[\ln p(X,\theta)] = \int \ln p(X,\theta) \prod_{i\neq j} q_i \text{d}\theta_i$
$q_j^\star = \frac{\exp(\mathbb{E}_{i\neq j}[\ln p(X,\theta)])}{\int \exp(\mathbb{E}_{i\neq j}[\ln p(X,\theta)]) \text{d}\theta_j}$
So in this case we maximize the bounds separately for each term w.r.t. to the others.
We can minimize two different kind of KL divergence: $\mathcal{KL}(q||p)$ which is the one used until now and the variance along the orthogonal direction is under-estimated; $\mathcal{KL}(p||q)$, which is used in Expectation Propagation, places significant probability mass in regions of variable space that have very low probability.

7 Sampling Methods

7.1 Rejection sampling

We take a $q(z)$ distribution, choose k for $kq(z) \geq p(z)$, $z \in p(z)$
First generate $z_0 \sim q(z)$, then $u_0 \sim U(0, kq(z_0))$ and finally if $u_0 \leq \tilde{p}(z_0)$ the sample (z_0) is accepted, rejected otherwise.
$p(accept) = \int \{p(z)/kq(z)\} q(z) \text{d}z = \frac{1}{K} \int p(z) \text{d}z$
Note that k has to be chosen so that kq is as close as possible to $\tilde{p}$. The sample will be rejected if it falls between kq and $\tilde{p}$, so for high-dimensional data the area of rejection will be too big.

7.2 Adaptive rejection sampling

When we have a distribution $p(z)$ where $\ln p(z)$ has derivatives which are not increasing functions of z. The envelope distribution is a succession of linear functions. $q(z) = k_i \lambda_i \exp\{-\lambda_i(z - z_i)\}$. Then rejection sampling can be applied.

7.3 Importance sampling

Provides a framework for approximating distributions but does not provide a mechanism for drawing samples from them. Used when it is not easy to find a function that forms an upper bound. Note that q does not necessarily have to be a bound on p, as we weigh by the quotient of the two distributions. Samples are drawn from distribution q and weighted correspondingly: $x_i \approx \tilde{q}$, $w_i = \frac{\tilde{p}(x_i)}{\tilde{q}(x_i)}$ Works bad in high dimensions as it is hard to find an x where both $q(x)$ and $p(x)$ are high in the high-dimensional space. If p and q overlap just a little, this will give a bad estimate.

7.4 Ancestral sampling (Likelihood Weighted sampling)

Given a Bayesian network, we can write down the joint probability as $p(z_1)p(z_2|z_1)$. That means there is always a node in the tree that is not dependent on any other variable. We sample this node, and use the sample to draw samples for subsequent nodes. If we observe values ($x \in Ev$, evidences), we just select the observed value instead of sampling. Works bad in high dimensions as well.

7.5 Markov Chain Monte-Carlo sampling

MCMC runs ancestral sampling on a chain. Samples can no longer be drawn independently. $t \to 1$, $x_t \sim q_\infty(x_\infty)$ Equilibrium or invariant distribution. We transition from q_1 to q_2 according to transition probability $T(x_2|x_1)$.
$q(x_{t+1}, x_t) = T(x_{t+1}|x_t) q(x_t)$
Invariance: We want to find T for a given p, such that $p = Tp$ holds.
Detailed balance: (or reversibility) We want to find T so that transition from one state to another has probability mass equal to the transition from that next state to the current. Thus, $p^*(z)T(z,z') = p^*(z')T(z',z)$. Distribution $p^*(z)$ is invariant if $p^*(z) = \sum_{z'} p^*(z')T(z',z)$
Proof: $A_k(z',z) = \min\left(1, \frac{p(z')q(z|z')}{p(z)q(z'|z)}\right)$
$p(z)q(z'|z)A_k(z',z) = \min\left(p(z)q(z'|z), p(z')q(z|z')\right)$ [multiply by denom]
$= \min\left(p(z')q(z|z'), p(z)q(z'|z)\right)$ [order irrelevant]
$= p(z')q(z|z')A_k(z,z')$
Ergodicity: This algorithm needs ergodicity: A positive probability for every state (so all will be reached).
For two transition kernels T_1, T_2 that are valid any linear combination of these two is a valid kernel.

7.6 Metropolis Algorithm

We assume that the proposal distribution is symmetric, $q(z_a|z_b) = q(z_b|z_a)$. As with rejection and importance sampling, we again sample from a proposal distribution. This time, however, we maintain a record of the current state $z^{(\tau)}$, and the proposal distribution $q(z|z^{(\tau)})$ depends on this current state, and so the sequence of samples $z^{(1)}, z^{(2)}, \ldots$ forms a Markov chain. At each iteration we generate a sample $z^\star$ directly from the proposal distribution and then accept according to
$A(z^\star, z^{(\tau)}) = \min(1, \frac{\tilde{p}(z^\star)}{\tilde{p}(z^{(\tau)})})$
This can be done by chosing a random number $u \sim U(0,1)$ and accepting if $u \leq A(z^\star, z^{(\tau)})$

7.7 Metropolis-Hastings Algorithm

Now the proposal distribution is no longer symmetric. The algorithm proceeds as normal MCMC but the acceptance probability is:
$A(z^\star, z^{(\tau)}) = \min(1, \frac{p(z^\star)q(z^{(\tau)}|z^\star)}{p(z^{(\tau)})q(z^\star|z^{(\tau)})})$ Metropolis-Hastings algorithm work better in high dimension than previous methods, but is quite slow (as we perform a random walk). So, if the algorithm satisfies *ergodicity* and *detailed balance*, it will eventually sample from the desired distribution $p(z)$. The first ("burn-in") samples should be discarded as they are not yet sampling from the equilibrium distribution. It is hard to say in general how many burn-in samples there are.

7.8 Gibbs sampling

For a distribution $p(\mathbf{z})$, for each component of the distribution z_i draw a sample from the distribution of that variable given the values of the remaining variables:

1. Initialize $z_i \forall i \in \{1,\ldots,M\}$
2. For $\tau = 1,\ldots,T$:
   - Sample $z_1^{\tau+1} \sim p(z_1|z_2^\tau,\ldots,z_M^\tau)$
   - Sample $z_j^{\tau+1} \sim p(z_j|z_1^{\tau+1},\ldots,z_{j-1}^{\tau+1},z_{j+1}^\tau,\ldots,z_M^\tau)$
   - Sample $z_M^{\tau+1} \sim p(z_M|z_1^{\tau+1},\ldots,z_{M-1}^{\tau+1})$

8 HMMs + Kalman filters

8.1 Markov models

First order Markov chains hold the Markov property which you whole history is only dependent on the current state. **First order**'s number of parameters will be: $k-1+k(k-1)$. **Second order**: $k^2-1+k^2(k-1)$.

8.2 HMMs

Useful in speech recognition, natural language processing, online character recognition... Discrete variables: HMM, Linear-Gaussian: LDS (Linear Dynamical System).

$p(x,\dots,x_n) = \sum_{z_1}\dots\sum_{z_n} p(z_1)(\prod_{n=2}^{N} p(z_n|z_{n-1},A))\prod_{n=1} p(x_n|z_n)$

For the first node:

$p(z_1|\pi) = \prod_{k=1}^{K} \pi_k^{z_{1,k}}$

$p(z_n|z_{n-1},A) = \prod_{k=1}^{K}\prod_{j=1}^{K} A_{jk}^{z_{n-1,j}z_{nk}}$

$p(x_n|z_n,\phi) = \prod_{k=1}^{K} p(x_n|\phi_k)^{z_{nk}}$

Where A, ϕ and π are the parameters of the model. $p(x_n|z_n,\phi)$ is model-dependent (e.g.: mixture of Gaussians, multinomial if discrete, etc).

8.2.1 Maximum likelihood (EM)

data: X, latent state: Z, parameters: $\theta = (\pi, A, \phi)$.

$p(x|\theta) = \sum_z p(x,z|\theta)$

This represents a sum of K^N terms, which quickly becomes intractable, so we can use for instance the EM algorithm.

E-step:

$Q(\theta|\theta^{old}) = \sum_z p(z|x,\theta^{old}) \ln p(x,z|\theta)$

M-step:

$\theta^{new} = \arg\max_\theta Q(\theta|\theta^{old})$

EM-general:

$\ln p(X|\theta) = \mathcal{L}(q,\theta) + \mathcal{KL}(q||p)$ where $\mathcal{L}(q,\theta) = \sum_Z q(Z)\ln(\frac{p(X,Z|\theta)}{q(Z)})$ and $\mathcal{KL}(q||p) = -\sum_Z q(Z)\ln(\frac{p(Z|X,\theta)}{q(Z)})$.

E-step: maximize $\mathcal{L}(q,\theta)$ wrt $q(Z)$. M-step: maximize $\mathcal{L}(q,\theta)$ wrt θ.

8.2.2 Forward-Backward

$a(z_n) = p(x_n|z_n)\sum_{z_{n-1}} a(z_{n-1})p(z_n|z_{n-1})$

$\beta(z_n) = \sum_{z_{n+1}} \beta(z_{n+1})p(x_{n=1}|z_{n+1})p(z_{n+1}|z_n)$

Predictive distribution: Add x_{n+1} and z_{n+1} at the end of the chain and use the calculated parameters to predict the new values.(assuming the chain is homogeneous). By d-separation (if applicable), $p(x_{N+1}|X) = p(x_{N+1}|x_N)$ (regular Markov Chain, add z's to taste.

8.3 Forward-backward for HMM

$p(x_N|X,z_{N+1}) = p(X_{N+1},z_{N+1})$

$p(z_{N+1}|z_N,X) = p(z_{N+1},z_N)$

$p(X|z_n) = p(x_1,\dots,x_n|z_n)p(x_{n+1},\dots,x_N|z_n)$

$\gamma(z_n) = p(z_n|X) = \frac{p(X|z_n)p(z_n)}{p(X)}$ [$p(X)$ implicitly conditioned on θ^{old}, is likelihood]

We can write $\gamma(z_n) = \frac{p(x_1,\dots,x_n,z_n)p(x_{n+1},\dots,x_N|z_n)}{p(X)} = \frac{\alpha(z_n)\beta(z_n)}{p(X)}$

Recursion relations:

$\alpha(z_n) = p(x_n|z_n)\sum_{z_{n-1}} \alpha(z_{n-1})p(z_n|z_{n-1})$

$\alpha(z_1) = p(x_1,z_1) = p(z_1)p(x_1|z_1) = \prod_{k=1}^{K}\{\pi_k p(x_1|\phi_k)\}^{z_{1k}}$

$\beta(z_n) = \sum_{z_{n+1}} \beta(z_{n+1})p(x_{n+1}|z_{n+1})p(z_{n+1}|z_n)$

$\beta(z_N) = 1$

$\xi(z_{n-1},z_n) = p(z_{n-1},z_n|X) = \frac{\alpha(z_{n-1})p(x_n|z_n)p(z_n|z_{n-1})\beta(z_n)}{p(X)}$

Joint $p(X) = \sum_{z_n} \alpha(z_n)\beta(z_n)$

EM:

E-step: Evaluate $\gamma(z_n)$ and $\xi(z_{n-1},z_n)$

M-step: maximize $Q(\theta,\theta^{old}) = \sum_Z p(Z|X,\theta^{old})\ln p(X,Z|\theta) = \sum_{k=1}^{K}\gamma(z_{1k})\ln\pi_k + \sum_{n=2}^{N}\sum_{j=1}^{K}\sum_{k=1}^{K}\xi(z_{n-1,j}z_{nk})\ln A_{jk} + \sum_{n=1}^{N}\sum_{k=1}^{K}\gamma(z_{nk})\ln p(x_n|\phi_k)$

8.4 Sum-product for HMM

We define:

$h(z_1) = p(z_1)p(x_1|z_1)$

$f_n(z_{n-1},z_n) = p(z_n|z_{n-1})p(x_n|z_n)$

Define final hidden variable z_n as root, and first pass from h to root:

$\mu_{z_{n-1}\to f_n}(z_{n-1}) = \mu_{f_{n-1}\to z_{n-1}}(z_{n-1})$

$\mu_{f_n\to z_n}(z_n) = \sum_{z_{n-1}} f_n(z_{n-1},z_n)\mu_{z_{n-1}\to f_n}(z_{n-1})$

We can set:

$\alpha(z_n) = \mu_{f_n\to z_n}(z_n)$

Next, we propagate from the root back to the leaf:

$\mu_{f_{n+1}\to f_n}(z_n) = \sum_{z_{n+1}} f_{n+1}(z_n,z_{n+1})\mu_{f_{n+1}\to f_{n+1}}(z_{n+1})$

And we can set:

$\beta(z_n) = \mu_{f_{n+1}\to z_n}(z_n)$

8.5 Viterbi for HMM

Again, set z_N as root and pass from leaf to root:

$\mu_{z_n\to f_{n+1}}(z_n) = \mu_{f_n\to z_n}(z_n)$

$\mu_{f_{n+1}\to z_{n+1}}(z_{n+1}) = \max_{z_n}\{\ln f_{n+1}(z_n,z_{n+1}) + \mu_{z_n\to f_{n+1}}(z_n)\}$

Eliminate $\mu_{z_n\to f_{n+1}}(z_n)$ and use $f_n(z_{n-1},z_n) = p(z_n|z_{n-1})p(x_n|z_n)$ to get a recursion:

$\omega(z_{n+1}) = \ln p(x_{n+1}|z_{n+1}) + \max_{z_n}\{\ln p(z_{n+1}|z_n) + \omega(z_n)\}$

These are initialized using:

$\omega(z_1) = \ln\, p(z_1) + \ln\, p(x_1|z_1)$

8.6 Linear Dynamical Systems

Transitions: $p(z_n|z_{n-1}) = \mathcal{N}(z_n|Az_{n-1},\Gamma)$ ($\Gamma \to$ transisition noise)

Observations: $p(x_n|z_n) = \mathcal{N}(x_n|Cz_n,\Sigma)$ ($\Sigma \to$ measurement noise)

Initial state: $p(z_1) = \mathcal{N}(z_1|\mu_0,V_0)$

Use EM to learn $A,\Gamma,C,\Sigma,\mu_0,V_0$

$\hat{\alpha}(z_n) = \mathcal{N}(z_n|\mu_n,V_n)$

$C_n\hat{\alpha}(z_n) = p(x_n|z_n)\int \hat{\alpha}(z_{n-1})p(z_n|z_{n-1})dz_{n-1}$

Using 2.115: $c_n = \mathcal{N}(x_n|A\mu_{n-1},\Sigma + CP_{n1}C^T)$

$\mu_n = (P_{n-1}^{-1} + C^T\Sigma^{-1}C)^{-1}(C^T\Sigma^{-1}x_n + P_{n-1}^{-1}A\mu_{n-1})$

$V_n = (P_{n-1}^{-1} + C^T\Sigma^{-1}C)^{-1}$

For stability:

$V_n = (I - K_nC)P_{n-1}$

$\mu_n = A\mu_{n-1} + K_n(x_n - CA\mu_{n-1})$ where $x_n - CA\mu_{n-1}$ is the error between observation and expected observation;

$c_{n=1}\hat{\beta}(z_n) = \int \hat{\beta}(z_{n+1})p(x_{n+1}|z_{n+1})p(z_{n+1}|z_n)dz_{n+1}$

$\hat{\mu}_n = \mu_n + J_n(\hat{V}_{n+1} - P_n)J_n^T$ where $J_n = V_nA^T(P_n)^{-1}$

Use $\gamma(z_n) = \hat{\alpha}(z_n)\hat{\beta}(z_n) = \mathcal{N}(z_n|\hat{\mu}_n,\hat{V}_n)$

Learning: $\ln p(X,Z|\theta) = \ln p(z_1|\mu_0,V_0) + \sum_{n=2}^{N}\ln p(z_n|z_{n-1},A,\Gamma) + \sum_{n=1}^{N}\ln p(x_n|z_n,C,\Sigma)$

$Q(\theta,\theta^{old}) = \mathbb{E}_{z|\theta^{old}}[\ln\, p(X,Z|\theta)]$

$\mathbb{E}[z_n] = \hat{\mu}_n$; $\mathbb{E}[z_nz_{n-1}^T] = \hat{V}_nJ_{n-1}^T + \hat{\mu}_n\hat{\mu}_{n-1}^T$; $\mathbb{E}[z_nz_n^T] = \hat{V}_n + \hat{\mu}_n\hat{\mu}_n^T$

Kalman gain matrix: K_n as found in $c_n = \mathcal{N}(x_n|\mu_{x_n},K_n)$

9 Mathematical Tricks

$-\frac{1}{2}(x-\mu)^\top\Sigma^{-1}(x-\mu) = -\frac{1}{2}x^\top\Sigma^{-1}x + x^\top\Sigma^{-1}\mu + c$

$\frac{\partial a^\top Xb}{\partial X} = ab^\top$, $\frac{\partial a^\top X^\top b}{\partial X} = ba^\top$

$\frac{\partial \det X}{\partial X} = \det(X)(X^{-1})^\top$

$M = M^\top \iff u^\top Mv = v^\top Mu$

$\mathbb{E}[(x-\mu)^\top\Sigma^{-1}(x-\mu)\mathcal{N}(x|\mu,\Sigma)] = \mathrm{Tr}(\Sigma^{-1}\Sigma) = D$

$\mathbb{E}[x] = \int_{-\infty}^{\infty} xf(x)\mathrm{d}x$

Product of Gaussians: $\Sigma = (\Sigma_1^{-1} + \Sigma_2^{-1})^{-1}$, $\mu = \Sigma\Sigma_1^{-1}\mu_1 + \Sigma\Sigma_2^{-1}\mu_2$

Other: $(P^{-1} + B^\top R^{-1}B)^{-1}B^\top R^{-1} = PB^\top(BPB^\top + R)^{-1}$

Woodbury: $(A + BD^{-1}C)^{-1} = A^{-1} - A^{-1}B(D + CA^{-1}B)^{-1}CA^{-1}$

Gradient in exp family wrt for η, for $p(x|\eta) = h(x)g(\eta)\exp\{\eta^Tu(x)\}$, where $g(\eta)\int h(x)\exp\{\eta^Tu(x)\}dx = 1$, we have the result:$-\nabla\ln\, g(\eta) = \mathbb{E}[u(x)]$