Machine Learning 2 - Cheat Sheet
Andrea Jemmett - April 18, 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
$X \perp\!\!\!\perp Y \mid Z \Longleftrightarrow p(X,Y|Z) = p(X|Z)p(Y|Z)$

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

2 Distributions

Binary	Bernoulli	Binomial	Beta
Discrete	Categorical	Multinomial	Dirichlet

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

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

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

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

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

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

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

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

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

[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] $B(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$

2.8.3 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.
$St(x|\mu,\lambda,\nu) = \frac{\Gamma(\nu/2+1/2)}{\Gamma(\nu/2)}(\frac{\lambda^{1/2}}{(\pi\nu)^{D/2}})[1+\frac{\lambda(x-\mu)^2}{\nu}]^{-\nu/2-D/2},$
$f_x(x) = \frac{\Gamma[(\nu+p)/2]}{\Gamma(\nu/2)\nu^{p/2}\pi^{p/2}|\Sigma|^{1/2}[1+1/\nu(x-\mu)^T\Sigma^{-1}(x-\mu)]^{(\nu+p)/2}}$
$\mathbb{E}(\mathbf{x}) = \frac{\Gamma(D/2+v/2)}{\Gamma(v/2)}\frac{|\Lambda|^{1/2}}{(\pi v)^{D/2}}\int[1+\frac{(x-\mu)^T\Lambda(x-\mu)}{v}]^{-D/2-v/2}x\text{d}x$

3 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 $\boldsymbol{\theta}$.

3.1 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 $\text{Beta}(\theta|a,b) \propto \theta^{a-1}(1-\theta)^{b-1}$. The **posterior** is then obtained by multiplying the prior with the likelihood, $p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta) = \text{Bin}(N_1|\theta,N_1+N_0)\text{Beta}(\theta|a,b) \propto \text{Beta}(\theta|a+N_1,b+N_0)$. 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 $p(\mathcal{D}) = \binom{N}{N_1}B(a+N_1,b+N_0)/B(a,b)$.

3.2 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 $p(\boldsymbol{\theta}|\mathcal{D}) \propto \text{Dir}(\boldsymbol{\theta}|\boldsymbol{\alpha})p(\mathcal{D}|\boldsymbol{\theta}) \propto \text{Dir}(\boldsymbol{\theta}|\alpha_1+N_1,\ldots,\alpha_k+N_k)$. The **evidence** is (obtained as the previous model) $p(\mathcal{D}) = B(\mathbf{N}+\boldsymbol{\alpha})/B(\boldsymbol{\alpha})$.

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

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

4.1.1 Blocking Rules
○ → ● → ○ ○ ← ● → ○ ○ → ⋯ ○ ⋯ ← ○

4.1.2 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.

4.2 Markov Random Fields
Graphical Models with undirected edges (a.k.a. Undirected Graphical Models).

4.2.1 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).

4.2.2 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.

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

4.2.4 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.

4.3 Markov Blanket
$p(x_i|x_{\text{MB}_i},x_{\text{rest}}) = p(x_i|x_{\text{MB}_i})$
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.

4.4 Naive Bayes
The problem of finding a label y^* for some previously unobserved vector of features $\mathbf{x}^*$, while having previously observed $\{\mathbf{x}_n,y_n\}_{n=1,\ldots,N}$. 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
$p(y,x_1,\ldots,x_D|\boldsymbol{\eta},\boldsymbol{\theta}_i) = p(y|\boldsymbol{\eta})\prod_{i=1}^D p(x_i|\boldsymbol{\theta}_i)$
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 $p(y|x_1,\ldots,x_D)$.
Finding a label y^* implies finding
$y^* = \text{argmax}_y\left\{\ln p(y|\boldsymbol{\eta}) + \sum_{i=1}^D \ln p(x_i^*|y,\boldsymbol{\theta}_i)\right\}$

4.5 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 $p(x_i|\text{pa}_{x_i})$.

$p(x_i|\text{pa}_{x_i}) = \frac{N(x_i,\text{pa}_{x_i})}{N(\text{pa}_{x_i})}$

is the number of times x_i co-occurred with pa_{x_i} divided by the number of times pa_{x_i} occurred.

4.6 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.

4.6.1 Inference on a chain

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

4.6.2 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.

4.6.3 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:

$$\mu_{f_\alpha\to x_i}(x_i) = \sum_{\mathbf{x}_\alpha\setminus x_i} f_\alpha(\mathbf{x}_\alpha) \prod_{m\in\text{ne}(f_\alpha)\setminus x_i} \mu_{x_m\to f_\alpha}(x_m)$$

variable → factor message:

$$\mu_{x_m\to f_\alpha}(x_m) = \prod_{\beta\in\text{ne}(x_m)\setminus f_\alpha} \mu_{f_\beta\to x_m}(x_m)$$

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: $p(x) = \prod_{\alpha\in\text{ne}(x)} \mu_{f_\alpha\to x}(x)$

factor marginal: $p(\mathbf{x}_\alpha) = f_\alpha(\mathbf{x}_\alpha) \prod_{i\in\text{ne}(f_\alpha)} \mu_{x_i\to f_\alpha}(x_i)$

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).

4.6.4 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:

$$\mu_{f_\alpha\to x_i}(x_i) = \max_{\mathbf{x}_\alpha\setminus x_i} \left[(\ln f_\alpha(\mathbf{x}_\alpha)) + \sum_{m\in\text{ne}(f_\alpha)\setminus x_i} \mu_{x_m\to f_\alpha}(x_m) \right]$$

variable → factor message:

$$\mu_{x_m\to f_\alpha}(x_m) = \sum_{f_\beta\in\text{ne}(x_m)\setminus f_\alpha} \mu_{f_\beta\to x_m}(x_m)$$

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 $p^{\max} = \max_x \left[\sum_{f_\alpha\in\text{ne}(x)} \mu_{f_\alpha\to x}(x)\right]$
$x^{\max} = \operatorname{argmax}_x \left[\sum_{f_\alpha\in\text{ne}(x)} \mu_{f_\alpha\to x}(x)\right]$

max-marginals $q(x_i) = -\ln Z + \sum_{f_\alpha\in\text{ne}(x_i)} \mu_{f_\alpha\to x_i}(x_i)$

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 $x_i^* = \operatorname{argmax}_{x_i} q(x_i)$.

4.6.5 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: $\forall C_i : \Psi_i := \prod_{\psi_j\in C_i}\psi_j$ $m_i \to C_j = \sum_{C_i\setminus S_{ij}} \Psi_i \prod_{k\in\text{ne}(i)\setminus j} m_{k\to i}(C_i)$

Note that: $C_i \setminus S_{ij} = C_i \setminus C_j$

Belief: $\Psi_{\text{root}}\prod_{k\in\text{ne(root)}} m_{k\to\text{root}}(C_{\text{root}}) \propto p(C_{\text{root}})$

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.

4.6.6 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*.