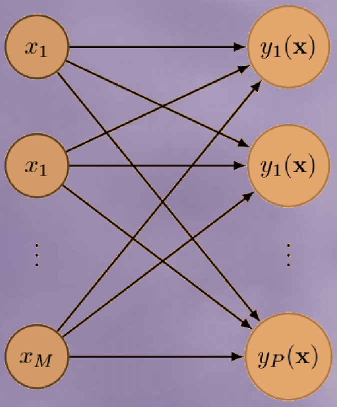
All of the Learning Algorithms

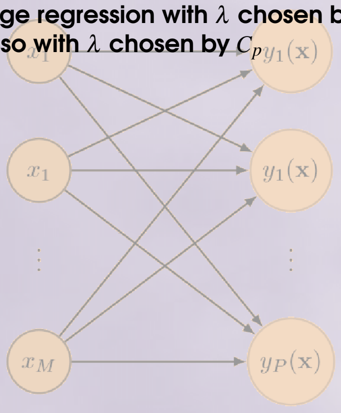
Daniel Khashabi

Versions 1.0

October 13, 2013



Contents



1 — Linear Models

1.1 Introduction

[TBW]

1.2 Simple linear regression model

Here we assume a linear model of regression. Lets consider an auxiliary matrix $\mathbf{X_{aux}} = [\mathbf{X}, 1]^T$. In least square regression we define the regressor as follows:

$$\mathbf{y} = \mathbf{X_{aux}}\beta + \varepsilon$$

Now it's clear that why we defined $\mathbf{X_{aux}}$. The reason is that we want our linear model has an intercept value. The goal is to minimize the following formula for RSS(residual sum of errors):

$$RSS = \sum_{i=1}^{N} (y_i - X_{aux,i}\beta)^2$$

It can be shown that the closed solution to this problem is as follows:

$$\hat{\beta} = \left(\mathbf{X_{aux}^T X_{aux}}\right)^{-1} \mathbf{X_{aux}^T y} \tag{1.1}$$

The linear least squares regresion is important in several ways. In addition to its simplicity, the Gauss-Markov Theorem asserts that the least squares estimate of the parameters β have the smallest variance among all linear unbiased estimates, though being "unbiased" doesn't mean to be perfect all the times, meaning that, there may exist a biased estimator that has smaller mean squares error. That is why Ridge regression and Lasso are introduced.

1.3 Subset selection based on AIC and BIC

With subset selection, we retain only a subset of predictor. So least squares is used to predict the result for variables that are retained.
In greedy stepwise *forward* subset-selection, we incrementally consider variables and add them up by considering the goodness criterion. While in *backward* selection we delete predictors that are redundant based on the information ciriteria. Though these criteria are sub-optimal,

there are various reasons that are prefered over other methods, especially from computational complexity point of view. Akaike Information Criterion(AIC) and BIC(Bayesian Information Criterion) are among measures for goodness of fit of a statistical model, to make a desired balance between bias and variance. *Backward* and *Forward* subset selection in R, use AIC and BIC as goodness measures.

1.4 Ridge regression with λ chosen by GCV

In order to add more continuity in regressions of linear least square method, we add a shrinkage parameter in the function to be minimized.

$$RSS = \sum_{i=1}^{N} \left(y_i - \beta_0 \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

By adding an extra parameter, λ which helps to regularize the sum of the errors. The closed solution to this minimization is as followoing :

$$\hat{\beta}^{ridge} = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

By changing the parameter λ we can get different regularized outputs of our model. For this model we define the *effective degree of freedom*, as a measure of how regularized the variables are, as follwing:

$$edf = \mathrm{tr}\left[\mathbf{X}\left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\right]$$

This criterion is reasonable; in fact when $\lambda = 0$, i.e. no regularization, and as a result $edf = p$. Whens $\lambda \to \infty$, i.e. full regularized regression, $edf \to 0$.

1.5 Lasso with λ chosen by C_p

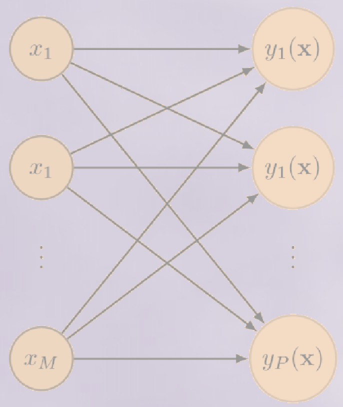
in *Lasso* model, similar to shrinkage mode, the estimate is given by:

$$RSS = \sum_{i=1}^{N} \left(y_i - \beta_0 \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

This model lies between the simple least squares model, and ridge shrinkage method. Unlike the ridge regression, there is no direct-way in Lasso to find the closed form for β.

In order to fine tune the parameter λ, the regularization parameter of Lasso, one can use Mallow C_p criterion, without cross-validation.

Ridge method, shrinks all of the predictors, but it shrinks low-variance predictors more, and keeps high-variance predictors(close to orthogonal). Lasso model acts somewhere between ridge regression and best-subset selection method. So it has some qualities of both models.



2 — Generalized Linear Models

2.1 Introduction

[TBW]

2.2 Generalized Regression Model

In boosting techniques the general idea is that *using a combination of several weak learner one could make a better learner*. One of the most famous method for such boosting based data is called “AdaBoost". Based on AdaBoost one could develop methods for gradient boosting, especially for trees. Here we first explain the AdaBoost.

Assume that we have a group of *weak* learners, $\{g_t(x)\}_{t=1}^T$, i.e. we have trained these learners on the training data $\{(x_i, y_i)\}_{i=1}^n$, which correspond to the weights $\{w_i\}_{i=1}^n$. We define the weighted output as $G(x) = \text{sign}\left(\sum_{t=1}^T \alpha_t g_t(x)\right)$.

Algorithm 1: Adaboost.

```
Input:
Output: The weighted output of weak learners, $G(x) = \text{sign}\left(\sum_{t=1}^T \alpha_t g_t(x)\right)$
Initialize the weights, $w_i^{(1)} = \frac{1}{n}$, $i = 1, \ldots, n$
for $t$ = 1 to $T$ do
    Fit the learner $g_t(x)$ to the training data $\{(x_i, y_i)\}_{i=1}^n$, weighted by $\{w_i\}_{i=1}^n$.
    $\varepsilon \leftarrow \sum_i w_i^{(t)} \mathbb{I}(y_i \neq g_t(x_i))$
    $\alpha_t \leftarrow \frac{1}{2} \log \frac{1-\varepsilon_t}{\varepsilon_t}$
    $w_i^{(t+1)} \leftarrow w_i^{(t)} \frac{\exp[-\alpha_t y_i g_t(x_i)]}{Z_t}$
Return the output $G(x)$.
```

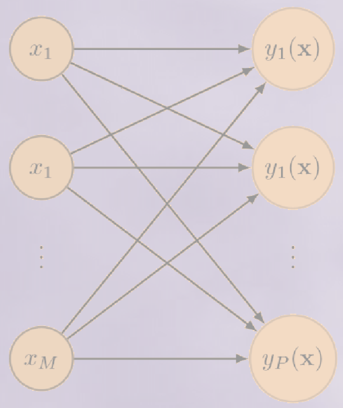
One could show that in AdaBoost the error of G_T is bounded by $\exp\left(-\sum_t \left(\frac{1}{2} - \varepsilon_t\right)^2\right)$. It can also be shown that if $\forall t,\ \varepsilon < 0.5$, error decreases.
Using the boosting methods, new methods are proposed for Gradient Boosting methods. In fact, the gradient boosting methods, are using both of *boosting* and *gradient* methods, especially gradient descent. Using only gradient methods have cones, e.g. negative effect on generalization error and local optimization. However, in glm suggests selecting a class of functions that

use the covariate information to approximate the gradient, usually a regression “tree". The algorithm is shown in Alg. 2. At each iteration the algorithm determines the direction, the gradient, in which it needs to improve the fit to the data and selects a particular model from the allowable class of functions that is in most agreement with the direction.

Algorithm 2: Gradient boosting for trees.

```
Input:
Output: The weighted output of weak learners, $G(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t g_t(x)\right)$
Initialize a single-node tree $f_0(x) = \arg\min_\gamma \sum_i L(y_i, \gamma)$.
for $m$ = 1 to $M$ do
    Compute the negative gradient, $-\mathbf{g}_m = (-g_{1m}, -g_{2m}, \dots, -g_{nm})^T$
    Fit the regression to the pseudo-responses $g_{im}$, given terminal regions,
    $R_{jm},\ j = 1, \dots, J_m$.
    For $j = 1, 2, 3, \dots, J_m$, compute $\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$
    Update, $f_m(x) = f_{m-1}(x) + \sum_j \gamma_{jm} I(x \in R_{jm})$
Return the output $f_M(x)$.
```

In this scenario the shrinkage parameter is the (or learning rate) parameter in gradient updates. So the main effort in variable selection is in selecting are the choice of n.trees and shrinkage parameters.



3 — Logistic regression

3.1 Introduction

[TBW]

3.2 Logistic regression

Logistic Regression is so similar to linear regression methods introduced, though its goal is to model *categorical* data. It combines *logistic function*,$\pi(x)$ with linear regression. By defining logistic function:

$$\frac{e^x}{1+e^x},$$

we define the structure of the *Logistic Regression*:

$$\pi(\mathbf{x}) = \frac{e^{\beta^{\mathbf{T}}.\mathbf{x}}}{1+e^{\beta^\top.\mathbf{x}}},$$

where the exponent term $\beta.\mathbf{x}$ is the linear combination of variables, with an intercept, the same as what we had in linear regression:

$$\beta^\top.\mathbf{x} = \beta_0^\top + \sum_{i=1}^{n} \beta_i^\top.x_i.$$

Or equivalently

$$\beta^\top.\mathbf{x} = \beta_0^\top + \sum_{i=1}^{n} \beta_i^\top.x_i = \ln\frac{\pi(\mathbf{x})}{1+\pi(\mathbf{x})}.$$

By observing that $0 \leq \pi(x) \leq 1$, we can assume that $\Pr(G=1|X=x) = \pi(x)$, which is two-category classification. We can similarly generalize the model to K-category class by defining needed equations:

$$\beta_{\mathbf{k}}^\top.\mathbf{x} = \beta_{k,0}^\top + \sum_{i=1}^{n} \beta_{k,i}^\top.x_{k,i} = \ln\frac{\Pr(G=k|X=x)}{1+\Pr(G=k|X=x)},\ 1 \leq k \leq K.$$

Now we classify the data $X = x$ using $G^* = \arg\max_k \Pr(G = k|X = x)$. Note that in the K-categoty case, we have $\sum_{i=1}^{n} \Pr(G = i|X = x) = 1$; thus we can write the following:

$$\Pr(G = k|X = x) = \frac{\exp\left(\beta_{\mathbf{k}}^{\top}.\mathbf{x}\right)}{1 + \sum_{k=1}^{K-1} \exp\left(\beta_{\mathbf{k}}^{\top}.\mathbf{x}\right)},\ 1 \leq k \leq K-1,$$

and

$$\Pr(G = K|X = x) = \frac{1}{1 + \sum_{k=1}^{K-1} \exp\left(\beta_{\mathbf{k}}^{\top}.\mathbf{x}\right)}.$$

Fitting the multinomial regression is strightforward using maximum-likelihood. By a little abuse of notation, we define $p_g(x;\Theta) = \Pr(G = g|X = x)$. We define the likelihood on N data points, as following:

$$l(\theta) = \sum_{i=1}^{N} \log p_{g_i}(x_i;\Theta),$$

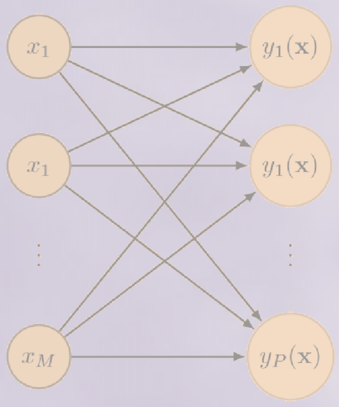
where Θ is the parameter matrix of the model. By having the training data-set, we can find the optimal parameters of the problem, by an interative maximization of the likelihood(ML),

$$\Theta^* = \arg\max_{\Theta} l(\theta) = \arg\max_{\Theta} \sum_{i=1}^{N} \log p_{g_i}(x_i;\Theta),$$

using Newton-method,

$$\Theta_{n+1} = \Theta_n - [Hl(\mathbf{x}_n)]^{-1} \nabla l(\mathbf{x}_n),\ n \geq 0.$$

One can derive the above gradient and Hessian matrix for the parameters of the problem, and train the system using the resulting *Iterative Reweighting Least Squares*.



4 — Discriminant Analysis

4.1 Introduction

[TBW]

4.2 Linear Discriminant Analysis

Discriminant Analysis methods are among methods for linear classification or the dimensionality reduction before classification method. In LDA we assume that conditional probability densities, $f(x|Y=y)$ are normally distributed. Let's assume that for classification posterior distribution we have the followings:

$$\Pr(G=k|X=x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l},$$

where π_ks are class membership priors. These priors show the probablity of membership in each of the classes, without having any training data at hand. Assuming multivariate Guassian distribution,

$$f_k(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}\left|\Sigma_k\right|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mu_k)^T \Sigma_k^{-1}(\mathbf{x}-\mu_k)}$$

The decision rule for deciding what class the data lies in is simply finding the maximum value of the class membership postorior:

$$G(x) = \arg\max_k \Pr\left(G=k|X=x\right). \tag{4.1}$$

In LDA we assume that all classes have the same covariance matrix $\Sigma_k = \Sigma_k$, $\forall k$. It can easily shown that one can write the following expression about the log-ratio of two classes,

$$\log \frac{\Pr\left(G=k|X=x\right)}{\Pr\left(G=l|X=x\right)} = \log \frac{\pi_k}{\pi_l} - \frac{1}{2}\left(\mu_k+\mu_l\right)^T \Sigma^{-1}\left(\mu_k-\mu_l\right) + \mathbf{x}^T\Sigma^{-1}\left(\mu_k-\mu_l\right), \tag{4.2}$$

which is the class margin, for classes l and k, when we have $\Pr\left(G=l|X=x\right) = \Pr\left(G=k|X=x\right)$. Based on the above equation we can easily see that the margin's equations are linear in terms of $\mathbf{x}$. That is why this method is called "Linear" Discriminant Analysis. By inspection it can be found that the decision rule in equation 4.1, can be equivalently written as the following equation:

$$G(x) = \arg\max_k \left\{ \mathbf{x}^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k \right\}.$$

Note that the approximate values of parameters for each class can be found using training data inside that class:

- Prior prbability of data, being in one class, can be estimated using the the number of data in that class over the number of the whole data: $\hat{\pi}_k = \frac{N_k}{N}$.
- Class distribution mean can be found by averaging over the data inside the class: $\mu_k = \frac{1}{N_k} \sum_{g_i=k} \mathbf{x}_k$.
- Class covariance can be found by calculating the covariance the data inside that class, but because be have assumed in the LDA that the covariance matrix of the all classes are the same, we average the covariance matrices over all the classes, which gives us the *pooled covariance matrix*: $\hat{\Sigma} = \frac{1}{N-K} \sum_{k=1}^{K} \sum_{g_i=k} (\mathbf{x}_i - \hat{\mu}_k)(\mathbf{x}_i - \hat{\mu}_k)^T$

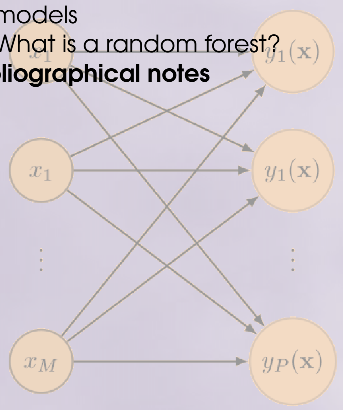
The cased of *Regularized Discriminant Analysis*(RDA) is a compromise between LDA and QDA. In fact by performing an averaging on LDA and QDA's covariance matrices, we shrink the covariance of QDA towards LDA's. We can define the regularized covariance matrix as following:

$$\hat{\Sigma}_k(\alpha) = \alpha \hat{\Sigma}_k + (1-\alpha) \hat{\Sigma}_k, \ 0 \geq \alpha \geq 1, \tag{4.3}$$

in which the parameter α controls the shrinkage level, and RDA's inclination towards LDA or QDA, and $\hat{\Sigma}_k$ is the *pooled covariance matrix*. Now the quadratic discriminant function can be defined using the shrunken covariance matrix $\hat{\Sigma}$. The optimum value of α could be evaluated using cross-validation on test data. Another modification to the regularized version would be to shrink the covariance matrix towards an scalar matrix, which is a constant times an identity matrix:

$$\hat{\Sigma}(\gamma) = \gamma \hat{\Sigma} + (1-\gamma) \hat{\sigma}^2 \mathbf{I}, \ 0 \leq \gamma \leq 1, \tag{4.4}$$

in which γ is a tuning parameter and could be determined using cross-validation on test data. To combine two regularization parameters, simply we replace $\hat{\Sigma}$ in 4.3 with $\hat{\Sigma}(\gamma)$, and name the result $\hat{\Sigma}(\alpha, \gamma)$.



5 — Tree Models

5.1 Introduction

[TBW]

5.2 Tree models

Assume input variables $\mathbf{x} = [\mathbf{x}_1, \ldots, \mathbf{x}_p] \in \mathscr{X}$, and output variables $\mathbf{y} \in \mathbb{R}$ or a categorical variable $\mathbf{y} \in \{1, \ldots, K\}$. The tree models are created by recursively splitting input space $\mathscr{X}$ on which the input variables are defined. Such recursive splitting creates partitions $\{R_n\}$ in the input space $\mathscr{X}$, and R_n defines the n-th region in the input space. We define the output prediction of the tree model as follows,

$$\hat{f}(\mathbf{x}) = \sum_n c_n I(\mathbf{x} \in R_n),$$

in which R_n is different input regions created by the tree on $\mathscr{X}$. For the case of classification problem, c_n is the *class label* (a categorical value), and for the case of regression it is usually a continuous value representing the *predicted output value*. In other words, label (value) of one given input, is the average of the labels(values) for any given input that belongs to the same region.

Now the problem is that how can we really create a desired suitable partitioning of the input space $\mathscr{X}$, in order to have an appropriate generalization(avoiding both underfitting and overfitting).
Let's say we have a tree structure and the partitioning created in the input space denoted by R_n. To minimize the error predictive and the real output, we want to minimize the following objective,

$$\min_{c_m} \sum_{m=1}^{M} \sum_{\mathbf{x}_i \in R_m} (y_i - c_m)^2.$$

By fixing the structure of the tree, the optimal value for each region R_m, is the mean of the values for that region, i.e. $c_m = \text{mean}(\{y_i | \mathbf{x}_i \in R_m\})$. To create a measure for splitting the branches, we need to define a criterion, namely *goodness of split function*, $\Phi(t, (i, j))$. For the case of regression we can use the following definition,

$$\Phi(t,(i,j)) = \text{RSS}(t) - [\text{RSS}(t_R) - \text{RSS}(t_L)], \; \text{RSS}(t) = \sum_{R_m \in \{R_i\}} \sum_{\mathbf{x}_i \in R_m} (y_i - c_t)^2 . \tag{5.1}$$

One important property for $\Phi(.)$ is that it is always positive. For the case of a categorical classification problem, for K categories, one could define it in a different way,

$$\Phi(t,(i,j)) = i(t) - [p_R.i(t_R) + p_L.i(t_L)], \; i(t) = I(\hat{p}_t(1), \dots, \hat{p}_t(K)), \tag{5.2}$$

in which $\hat{p}_t(j)$ is the impurity measure of the class j at node t and $\sum_j p_j = 1$. The function $I(.)$ is called *impurity measure*, and has it's maximum value when $p_1 = p_2 = \dots = p_K = \frac{1}{K}$. It is minimum when $p_j = 1$. A popular example of impurity measures is *entropy measure*, $\sum_{j=1}^{K} p_j(1-p_j)$.
Generally, one would prefer to grow the tree as much as possible and then prune it using *cost-complexity measure*,

$$R_\alpha(T) = R(T) + \alpha|T|.$$

In the above formulation, $R(.)$ is the error term for tree. In the regression defined as eq. 5.1, and in classification defined as eq. 5.2. The parameter α is the *relaxation* parameter. We prune the tree based the *relaxed* criterion, which possibly increase the error on the training data, while gaining more generalization over unseen data.

One unanswered question is the choice of α. One may follow different tastes for this choice, e.g. AIC, BIC, cross-validation.

5.2.1 Advantages and disadvantages of tree models

Tree models bring some advantages over the other models. These models are called *non-parametric* since the tree structure is created only based on the input data, and parameters of the information gains. So the tree is implicitly created by the data, and is not manually engineered. This also makes the interpretation of the structures easier. By cleverly tuning the splitting criterion one can make it robust to outliers.

5.2.2 What is a random forest?

In*Random Forest* the goal is to grow a group of trees, to decrease variance in the output and increase the certainty and stability of regression/classification done using the conventional tree models. In order to do so, we use the method of *Bootstrap Aggregation* or simply *Bagging*. In Bagging, instead of using all training data for creating the model, we choose only a subset of the data, to create a model. We then consider the result of aggregation among trained function on bootstrap samples, which is more consistent. The algorithm is shown in Alg. 3.

Algorithm 3: Bootstrap Aggregation or Bagging.

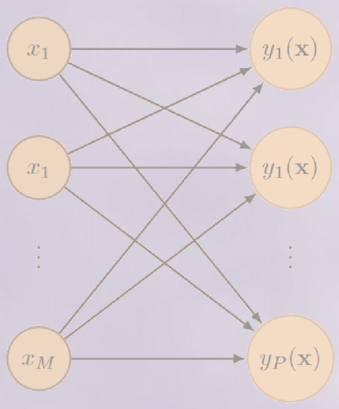
```
Input: Training data $\mathscr{D} = \{(x_i, y_i)\}_{i=1}^{N}$
Output: Regression/classification, $\hat{f}_{\text{bag}}(x)$
Generate random Bootstrap samples, $\mathscr{D}^b = \{(x_i^b, y_i^b)\}_{i=1}^{M}$, where $b = 1, \dots, B$.
Train $\hat{f}_{\text{bag}}^b$ using $\mathscr{D}^b$.
For regression: Averaging: $\frac{1}{B}\sum_{b=1}^{B} \hat{f}_{\text{bag}}^b$.
For classification: Majority voting among $\hat{f}_{\text{bag}}^1, \dots, \hat{f}_{\text{bag}}^B$.
```

Based on the idea of Bagging, we train tree model on each of the Bootstrap samples. I could assume various restrictions while growing tree models. One idea would be to randomly select m variables from the p variables, and then pick the best split among them. Selecting sensible

values for m could reduce the correlation between trees in the forest.
One good idea to find an estimate of error is to use those sample point that are not included in $\mathscr{D}^b$; these sample points are called Out-Of-Bag (OOB) samples.
One could calculate an estimate of *variable importance* for each variable, using Gini index, which is the splitting criterion. Improvement in the split-criterion (e.g. Gini index) is the importance measure attributed to the variable. The importance measure is the accumulated improvement in the split-criterion over all the trees in the forest for each variable.
Other than using the above method for variable importance one could use shuffling the values of the m-th variable inside the OOB samples, and then calculating the difference of the number of correctly classified before and after this change; repeat this work over all of the trees and average the overall result.

5.3 Bibliographical notes

In preparation of this document I have used lecture notes for STAT:542 at UIUC by Feng Liang.



6 — Expectation Maximization

6.1 Introduction

Consider the problem of parameter learning by maximizing the likelihood of the observations for random variables $(X,Y) \sim \{(x_i,y_i)\}_{i=1}^n$. Assume we model the joint distribution between X and Y using $p(X,Y;\theta)$ which is resulted designer's domain knowledge, and is characterized by the parameter θ.

$$\mathscr{L}(\theta) = \log \prod_{i=1}^{n} p(X,Y;\theta) = \sum_{i=1}^{n} \log p(X,Y;\theta)$$

To find the ML estimated parameters, one can maximize $\mathscr{L}(\theta)$ with respect to the model parameters θ. But what if we don't observe anything from Y? We call this scenario the "missing data" case. Assume the following definition of the likelihood,

$$\mathscr{L}(\theta) = \log \prod_{i=1}^{n} p(X;\theta) = \log \prod_{i=1}^{n} \sum_{Y} p(X,Y;\theta) = \sum_{i=1}^{n} \log \sum_{Y} p(X,Y;\theta)$$

$$\mathscr{L}(\theta) = \sum_{i=1}^{n} \log \sum_{Y} p(X,Y;\theta) \tag{6.1}$$

Performing parameter learning in the case of missing data (latent variables) by maximizing 6.1 is not trivial. But EM introduced a formalized way to approximate the maximization, by maximizing the lower-bound on this function.

6.2 Expectation-Maximization

We first introduce the algorithm, and then analyze its properties. The EM algorithm is the following,

Theorem 6.1 — The EM algorithm. Performing the following iterative steps, will result on the local maximizer of Equation 6.1.

- **Initialization:** Initialize the parameters of the mode θ.
- Repeat until convergence:

1. **Expectation:** Find the expected likelihood, $\mathscr{L}_{EM}$.

$$\mathscr{L}_{EM}(\theta;\theta_n) = \mathbb{E}_{\mathbf{Y}|\mathbf{X},\theta_n}\left[\log p\left(\mathbf{X},\mathbf{y}|\theta\right)\right] \tag{6.2}$$

2. **Maximization:** Maximize the EM objective $\mathscr{L}_{EM}$ with respect to θ.

We prove this iterations will converge to the maximization of Equation 6.1 in several steps.

■ **Lemma 6.1** The EM objective in Equation 6.2 is a lower bound on Equation 6.1.

$$\mathscr{L}_{EM}(\theta;\theta_n) \leq \mathscr{L}(\theta)$$

Proof. TODO 1 ■

[More explanation and analysis here]

■ **Example 6.1 — A simple Bayesian network(from Ran Roth).** Assume that a set of 3-dimensional points (x,y,z) is generated according to the following probabilistic generative model over Boolean variables $X,Y,Z \in \{0,1\}$:

$$Y \leftarrow X \rightarrow Z$$

1. What parameters from the table bellow will you need to estimate in order to completely define the model?

(1) `P(X=1)`	(2) `P(Y=1)`	(3) `P(Z=1)`	
(4) `P(X|Y=b)`	(5) `P(X|Z=b)`	(6) `P(Y|X=b)`	(7) `P(Y|Z=b)`
(8) `P(Z|X=b)`	(9) `P(Z|Y=b)`	(10) `P(X|Y=b,Z=c)`	(11) 3

Answer: Based on the above generative model we could write the joint distribution as following:

$$p(X,Y,Z) = p(X).p(Y|X).p(Z|X).$$

So we need to have (1), (6), (8). For this problem in order to find the whole joint distribution we need to know five parameters. For simplicity we denote the parameters using the following:

$$p(X=1) = \alpha$$
$$p(Y=1|X=1) = a_1$$
$$p(Y=1|X=0) = a_2$$
$$p(Z=1|X=1) = b_1$$
$$p(Z=1|X=0) = b_2$$

Then we have:

$$\begin{aligned}
p(X=x) &= \alpha^x(1-\alpha)^{1-x} \\
p(Y=y|X=1) &= a_1^y(1-a_1)^{1-y} \\
p(Y=y|X=0) &= a_2^y(1-a_2)^{1-y} \\
p(Z=z|X=1) &= b_1^z(1-b_1)^{1-z} \\
p(Z=z|X=0) &= b_2^z(1-b_2)^{1-z}
\end{aligned}$$

2. You are given a sample of m data points sampled independently at random. However, when the observations are given to you, the value of X is always omitted. Hence, you get to see $\{(y^1,z^1),\dots,(y^m,z^m)\}$. In order to estimate the parameters you identified in part (a), in the course of this question you will derive update rules for them via the EM algorithm for the given model.
Express $\Pr(y^j,z^j)$ for an observed sample (y^j,z^j) in terms of the unknown parameters.
Answer: We can use the joint distribution and integrate out the unseen variables:

$$\begin{aligned}
Pr(y^j,z^j) &= \sum_i Pr(X=x^i,y^j,z^j) \\
&= \sum_i p(X=x^i).p(Y=y^j|X=x^i).p(Z=z^j|X=x^i)
\end{aligned}$$

We can replace each term with its own parameter denoted in the previous part. - Let $p_i^j = Pr(X{=}i|y^j,z^j)$ be the probability that hidden variable X has the value $i \in \{0,1\}$ for an observation (y^j,z^j), $j \in \{1,\dots,m\}$. Express p_i^j in terms of the unknown parameters.
Answer: Using the Bayes law we could express the probability like this:

$$\begin{aligned}
p_i^j = Pr(X{=}i|y^j,z^j) &= Pr(y^j,z^j|X{=}i).Pr(y^j,z^j)/Pr(X{=}i) \\
&= Pr(y^j|X{=}i).Pr(z^j|X{=}i).Pr(y^j,z^j)/Pr(X{=}i)
\end{aligned}$$

Each of the terms are previously calculated.

3. Let (x^j,y^j,z^j) represent the completed j^{th} example, $j \in \{1,\dots,m\}$. Derive an expression for the expected log likelihood (LL) of the completed data set $\{(x^j,y^j,z^j)\}_{j=1}^m$, given the parameters in (a).
Answer:
$LL = \sum_j \log p(x^j,y^j,z^j) = \sum_j \log p(X=x^j) + \sum_j \log p(Y=y^j|X=x^j) + \sum_j \log p(Z=z^j|X=x^j)$ - Maximize LL, and determine update rules for <u>any two</u> unknown parameters of your choice (from those you identified in part (a)).
Answer: Because we don't haven't seen the latent variable values x we cannot explicitly plug in their values into the log-likelihood. But instead we need to takes its expectation with respect to the variable. This corresponds to the E-step in EM algorithm:

$$Q(y,z) = \sum_x p(x|y,z)\log p(x,y,z)$$

In the next step we shall maximize the expected likelihood with respect to the parameters:

$$\theta = \arg\max_\theta Q(y,z)$$

We can calculate the expectation of the whole-data likelihood as follows:

$$Q = \mathbb{E}\left[\sum_j \log p(x^j, y^j, z^j)\right] = \sum_j \mathbb{E}\left[\log p(x^j, y^j, z^j)\right] = \sum_j [p_1^j A + p_0^j B]$$

Where,

$$A = y^j \log b_1 + (1-y^j)\log(1-b_1) + z^j \log a_1 + (1-z^j)\log(1-a_1) + \log\alpha$$

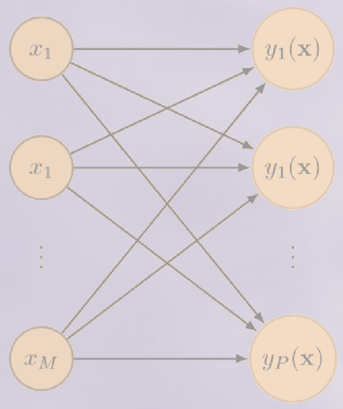
$$B = y^j \log b_2 + (1-y^j)\log(1-b_2) + z^j \log a_2 + (1-z^j)\log(1-a_2) + \log(1-\alpha)$$

To maximize the above expression, we must take derivitive with respect to the parameters:

$$\frac{\partial Q}{\partial \alpha} = \sum_j p_1^j \frac{1}{\alpha} - \sum_j p_0^j \frac{1}{1-\alpha} = 0 \Rightarrow \alpha = \frac{\sum_j p_1^j}{\sum_j p_1^j + \sum_j p_0^j}$$

$$\frac{\partial Q}{\partial b_1} = \sum_j p_1^j y^j \frac{1}{b_1} - \sum_j p_1^j (1-y^j) \frac{1}{1-b_1} = 0 \Rightarrow b_1 = \frac{\sum_j p_1^j y^j}{\sum_j p_1^j}$$

■



7 — Clustering Methods

7.1 K-means and Silhouette Statistic:

The k-means clustering method is described in the previous section. So I don't bring the explanations here again. The silhouette statistic of a measurement shows how well it fits in its own cluster versus how well it fits in its next closest cluster. The silhouette value for i-th measurement could be calculated by

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i) - b(i)\}}$$

Based on this formula, $-1 \leq s(i) \leq 1$, while generally $s(i) \geq$ and slightly around zero. As $s(i)$ gets closer to one, the better clustering is done. The "Silhouette" test uses the silhouette statistic to test how well data is clustered. The silhouette test subdivides the data into successively more clusters looking for the first minimum of the silhouette statistic. The silhouettes for k-means with different k values in Fig. 7.1. Based on the this figure, we choose $k = 2$.

The silhouette diagram for clustering for $k = 2$ is shown in Fig. 7.2. While $k = 2$ is the best value for clustering, the data is so intertwined and not so appropriate for fully clustering. That's why the silhouette for a big proportion of data is negative. In order to visualize the results of the clustering we use PCA to decompose the data into various dimension. The Fig 7.3 shows the proportion of eigenvalues(variances). In demonstrations we use 8 first PCs. The result of pairwise visualization is shown in Fig. **??** Note that we use this pairwise visualization because the data in different dimensions is intertwined. So we need to see the clustering from different directions.

The clustering figure in the pairwise in Fig. **??** also show how the data is intertwined

7.1.1 Hierarchical Clustering:

In hierarchical clustering we make use of mutual dissimilarity measure to build a hierarchical clustering structure. It this method we only need pairwise dissimilarity. To create this hierarchical clustering structure, we can follow different paths. One possible way is to do bottom-up clustering, in which we assume the data points to be in different clusters and we gradually merge them together until all observations are in one unit cluster. In *top-down clustering* we start with one cluster and gradually split them. In order to define a dissimilarity between two clusters each with more than one elements, we could use different approaches. For example *Single-linkage* assumes the nearest neighbours in each groups. In *complete-linkage* we use furthest neighbours

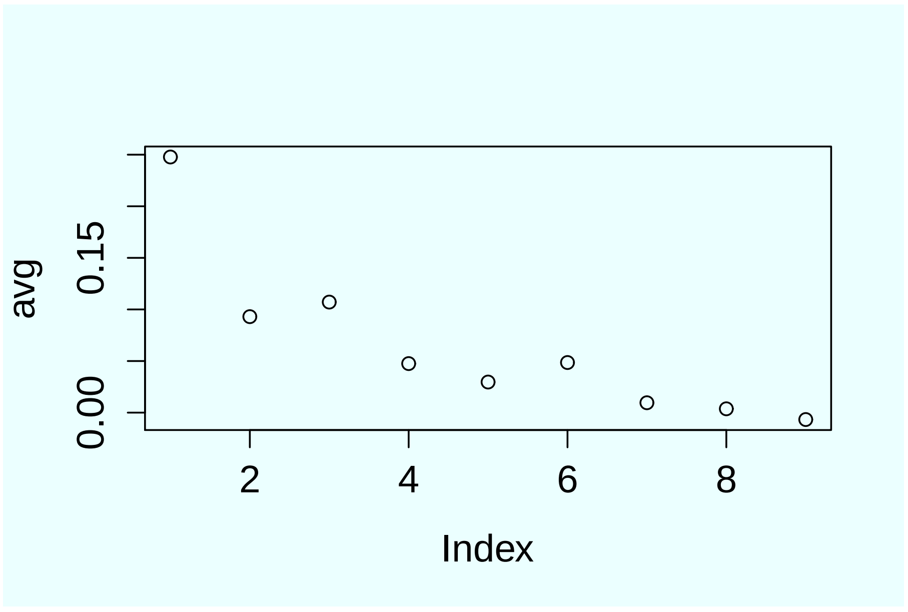


Figure 7.1: The silhouettes for k-means with different *k* values.

from each group. Another reasonable way is to assume *group average*. There also some other methods for doing this comparison.

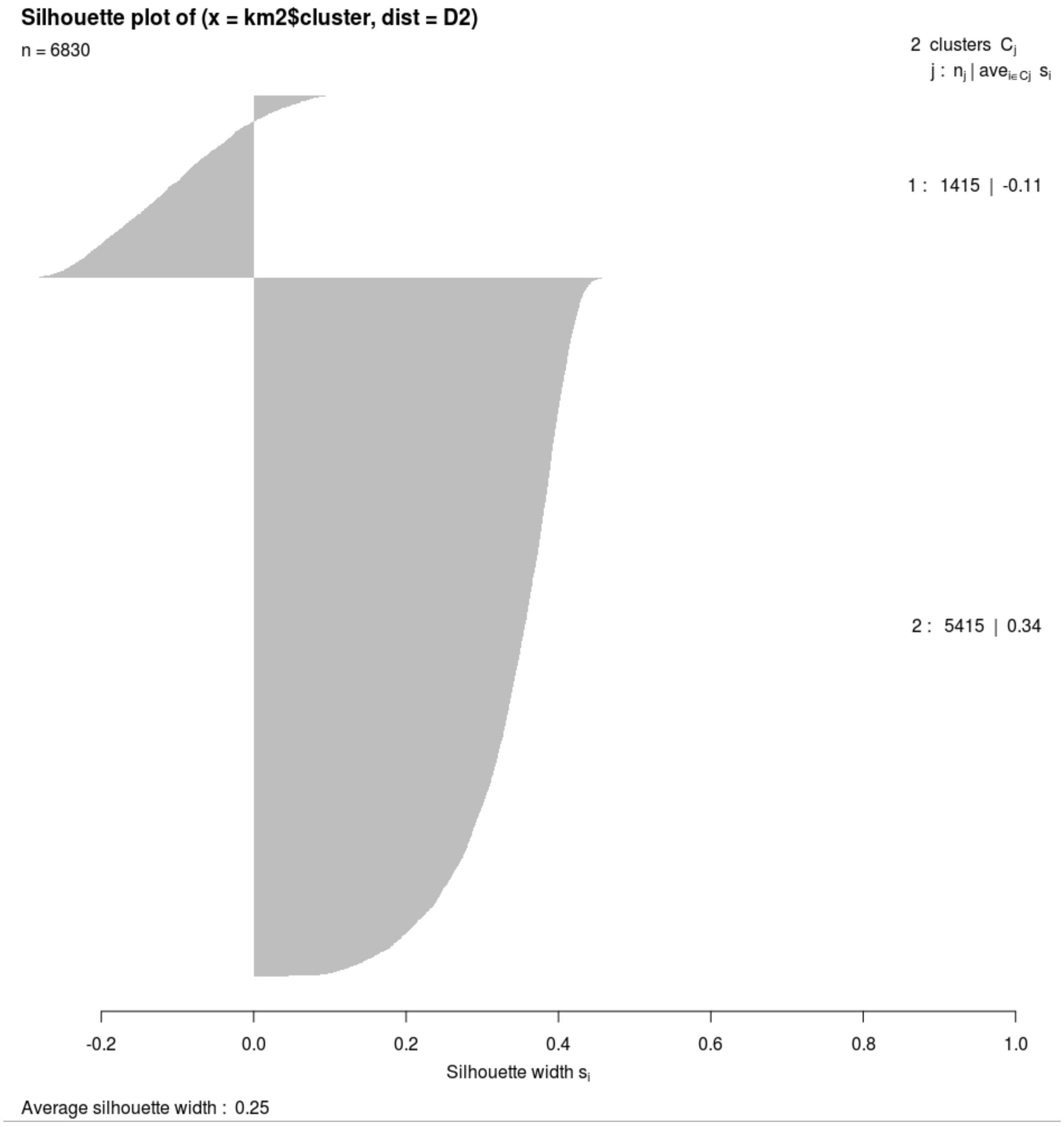


Figure 7.2: The silhouette diagram of k-means clustering for $k = 2$.

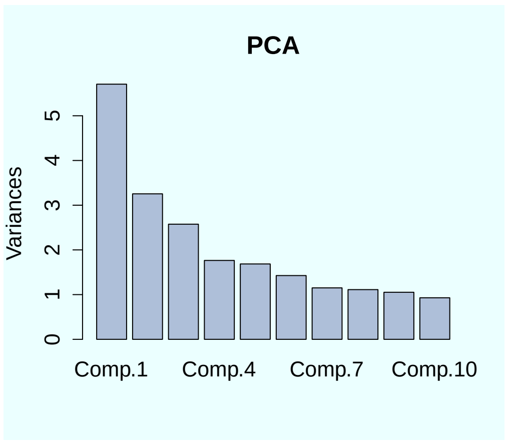


Figure 7.3: Variance portion of all eigenvalues of major directions.

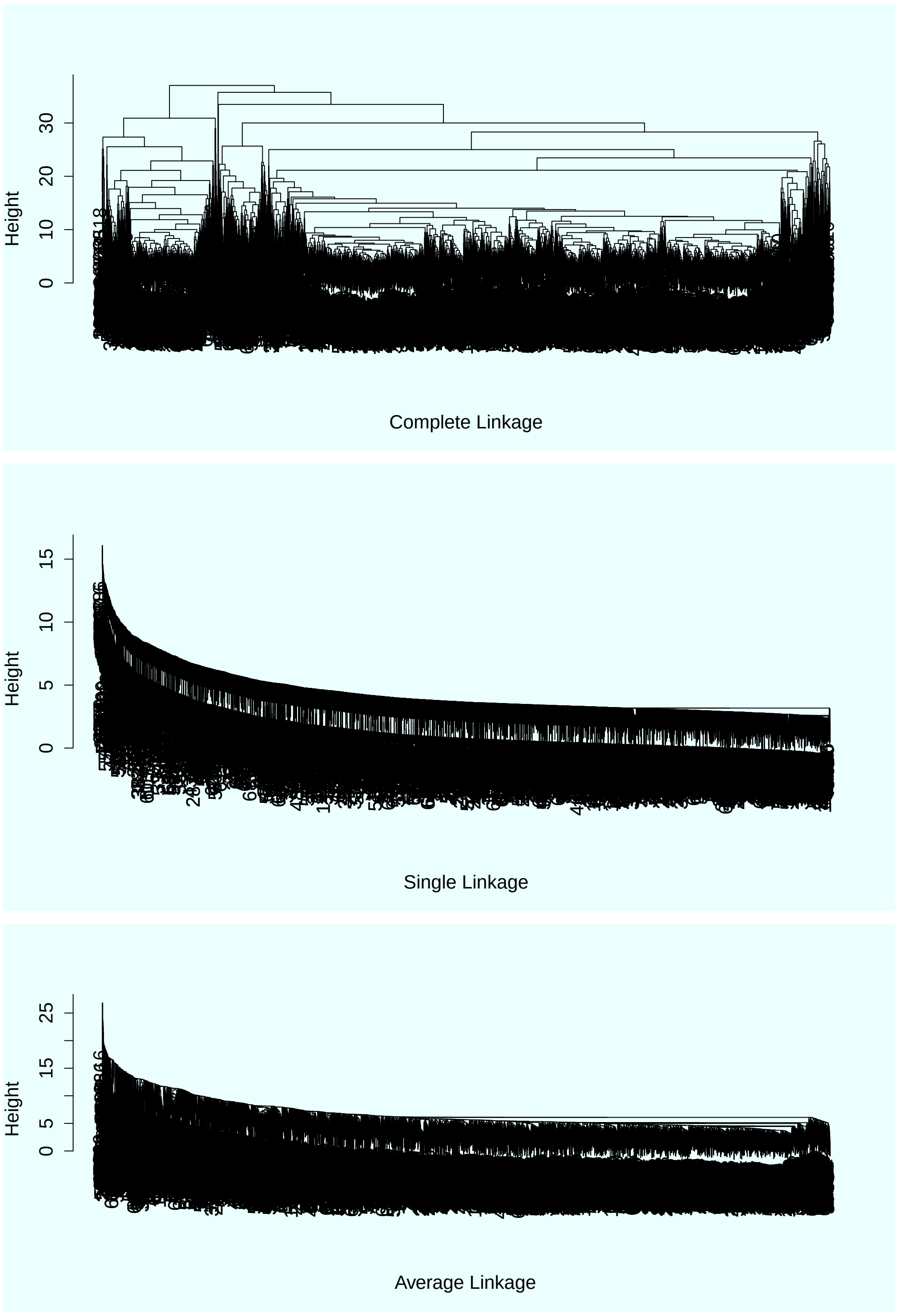
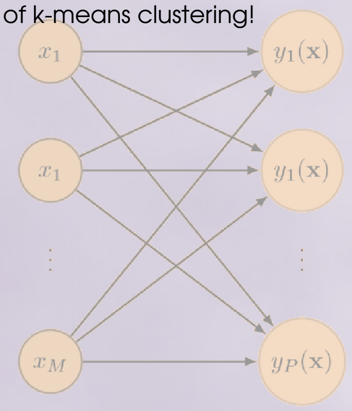


Figure 7.4: The various clustering methods in hierarchical clustering.



8 — Mixture Models

8.1 Introduction

[TBW]

8.2 EM for a mixture model

We want to train a Gaussian mixture model using EM. Let assume we have this mixture model:

$$g(x) = \sum_{k=1}^{K} \pi_k g_k(\mathbf{x}),\ g_k = \mathcal{N}\left(\mu_k, \sigma^2 \mathbf{I}\right)$$

such that $\sum_{i=1}^{K} \pi_k = 1$ and $\Theta = \left\{ \{\pi_k, \mu_k\}_{k=1}^{K}, \sigma^2 \right\}$ are unknown variables. Assuming that we have the training data $\{\mathbf{x}_n, g_n\}_{i=1}^{n}$, we can write the likelihood as following:

$$\mathcal{L} = \log \prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\mathbf{x}_i | \mu_k, \sigma^2 \mathbf{I}\right) = \sum_{i=1}^{n} \log \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\mathbf{x}_i | \mu_k, \sigma^2 \mathbf{I}\right). \tag{8.1}$$

Now we could find the MLE estimation of the parameters using gradient with respect to parameters of the model:

$$\frac{\partial \mathcal{L}}{\partial \pi_k} = 0,\ \frac{\partial \mathcal{L}}{\partial \mu_k} = 0,\ \frac{\partial \mathcal{L}}{\partial \sigma^2} = 0.$$

Because taking the derivatives of the likelihood in Eq. 8.1 is hard, we could change its form by adding a categorical latent variables, $\{z_k\}_{k=1}^{n}$ $s.t.$ $z_k = 0, \ldots, K$, that determine each of the samples come from which component:

$$z_i \sim \text{Cat}(K, p)$$

$$\mathbf{x}_i | z_i = k\ \sim \mathcal{N}\left(\mu_k, \sigma^2 \mathbf{I}\right)$$

$$\mathcal{L} = \sum_{k=1}^{K} \sum_{i:z_i=k}^{n} \log \mathcal{N}\left(\mathbf{x}_i | \mu_k, \sigma^2 \mathbf{I}\right) + \log \pi_k.$$

If we know z_i, the MLE estimation for each of the parameters could be found by

$$\hat{\pi}_k = \frac{n_k}{n},\ \hat{\mu}_k = \frac{1}{n_k} \sum_{i:z_i=k} \mathbf{x}_i,\ \hat{\sigma}^2 = \frac{1}{n} \sum_i \left(\mathbf{x}_i - \hat{\mu}_i\right)^T \left(\mathbf{x}_i - \hat{\mu}_i\right).$$

Now based the values of the model parameters we can calculate the probability of $\mathbf{x}_i$ belonging to one the component k by

$$\gamma_{ik} = \Pr(z_i = k|\mathbf{x}_i, \theta) = \frac{\pi_k \mathcal{N}\left(\mathbf{x}_i|\mu_k, \sigma^2\mathbf{I}\right)}{\sum_{k'=1}^{K} \pi_{k'} \mathcal{N}\left(\mathbf{x}_i|\mu_{k'}, \sigma^2\mathbf{I}\right)}. \tag{8.2}$$

Using the above criterion we can modify the likelihood updates as

$$\hat{\pi}_k = \frac{\sum_{i=1}^{n} \gamma_{ik}}{n}, \; \hat{\mu}_k = \frac{\sum_{i=1}^{n} \gamma_{ik}\mathbf{x}_i}{\sum_{i=1}^{n} \gamma_{ik}}, \; \hat{\sigma}^2 = \frac{\sum_i \gamma_{ik} \left(\mathbf{x}_i - \hat{\mu}_i\right)^T \left(\mathbf{x}_i - \hat{\mu}_i\right)}{\sum_{i=1}^{n} \gamma_{ik}} \tag{8.3}$$

One important question is that how to initialize the model parameters? A good way to construct initial guesses for μ_1 and μ_2 is simply to choose K of the training data at random. For σ^2 we can set it equal to the sample variance $\sum_{i=1}^{N} \left(\mathbf{x}_i - \hat{\mathbf{x}}\right)^2 / N$, and the mixing proportions, $\pi_k = 0.5$.

8.2.1 Gaussian Mixture Model as special case of k-means clustering!

First we have a short review on k-means clustering. In k-means clustering algorithm we aim to partition $X = \{x_1, \ldots, x_n\}$ into k clusters. Thus we define $\{\mu_i\}_{i=1}^{K}$ as centre of clusters, and $\left\{\{r_{i,k}\}_{i=1}^{n}\right\}_{k=1}^{K}$ as indicator variables. Each $r_{i,k}$ is 1 if and only if, x_i belongs to cluster k. The goal of the clustering is to minimize the sum of distances of points in the same cluster from the mean of the cluster:

$$J(\mathbf{r}, \mu) = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{i,k} \|\mathbf{x}_i - \mu_j\|^2$$

It can be shown that iterative repetition of the following two steps will result in the above objective function:

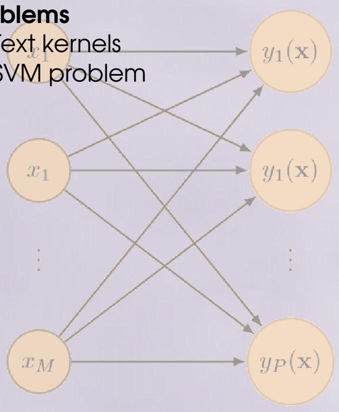
Step1: Assuming μ is determined, we can find $\mathbf{r}$ by

$$r_{i,k} = 1 \text{ if } k = \arg\min_j \|\mathbf{x}_i - \mu_j\|^2$$

Step2: Assuming $\mathbf{r}$ is fixed, we can find each centre of cluster by

$$\mu_k = \frac{\sum_{i=1}^{n} r_i \mathbf{x}_i}{\sum_{i=1}^{n} r_i} \tag{8.4}$$

If we assume that $\sigma \to 0$, the Gaussian distribution becomes one infinite mass at mean. So $\gamma_{i,k}$ in Eq. 8.2 becomes 1 only for $\mathbf{x}_i$ which is closest to μ_k, which is like $r_{i,k}$ in k-means clustering. Consequently Eq. 8.3 reduces to Eq. 8.4, and in overall results in k-means clustering algorithm.



9 — Support Vector Machines

9.1 Introduction

[TBW]

9.2 Simple classification using Support Vector Machines

In *Support Vector Machine*(SVM) we aim to choose a group of data points, which could reasonably separate information regions. Those data points that lie close to separation regions, selected among all the input data, are called "support vectors". Assume that we have group of data $\{\mathbf{x}_i, y_i\}$that could be separated by a hyperplane. Thus we can write the following statements about the separating hyperplanes,

$$\begin{cases} \beta.\mathbf{x}_i + \beta_0 \geq +1, & \text{if } y_i = +1 \\ \beta.\mathbf{x}_i + \beta_0 \leq -1, & \text{if } y_i = -1. \end{cases}$$

Equivalently we could write the above separating equations as follows:

$$y_i.(\beta.\mathbf{x}_i + \beta_0) \geq 0, \; \forall.$$

We define the optimality criterion for separation of two hyperplanes, as following:

$$\begin{cases} \min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2, \\ \text{constraints: } y_i.(\beta.\mathbf{x}_i + \beta_0) - 1 \geq 0 \end{cases}$$

The minimization criterion, $\min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2$, makes the coefficient vector, β, while preserving the separation constraints. This makes the coefficients to become as *sparse* as possible, based on *the defined criterion*. One other interpretation for the above optimization criterion is as follows; assume the Fig. 9.1. Consider two data points on the margin of each area, $(\mathbf{x}_1, +1)$ and $(\mathbf{x}_2, -1)$. For these two data points we have,

$$\begin{cases} \beta.\mathbf{x}_1 + \beta_0 = +1, & \text{if } y_1 = +1 \\ \beta.\mathbf{x}_2 + \beta_0 = -1, & \text{if } y_2 = -1. \end{cases} \Rightarrow \beta.(\mathbf{x}_1 - \mathbf{x}_2) = 2 \Rightarrow \|\mathbf{x}_1 - \mathbf{x}_2\| = \frac{2}{\beta}$$

So in order to maximize the separating margin $\|\mathbf{x}_1 - \mathbf{x}_2\|$ between data points, it suffices to minimize $\|\beta\|$ or minimize $\|\beta\|^2$. Minimizing $\|\beta\|^2$ is easier because of practical optimization

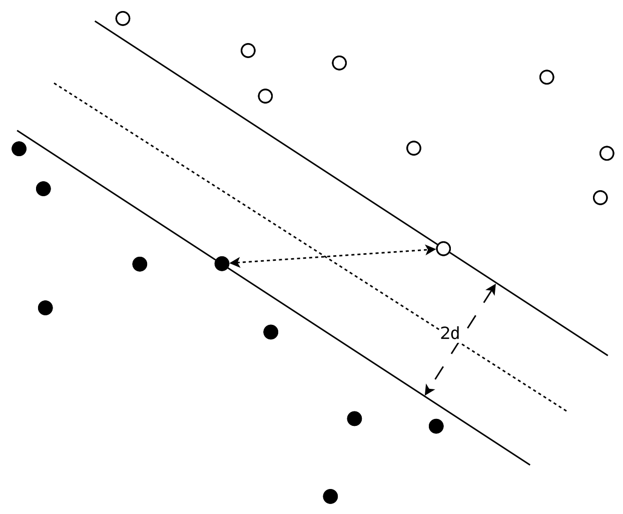


Figure 9.1: Max-margin scheme for support vector classification.

issues which makes the desired program convex, thus easier to handle. One can minimize the above defined convex program using KKT method. Now we can define the *Lagrange function* as following

$$L = \frac{1}{2}||\beta||^2 - \sum_i \lambda_i y_i (\mathbf{x}_i \beta + \beta_0) + \sum_i \lambda_i. \tag{9.1}$$

The above Lagrange function satisfies the needed conditions,

$$\begin{cases} \nabla_\beta L = 0 & \Rightarrow \beta = \sum_i \lambda_i y_i \mathbf{x}_i, \\ \frac{\partial L}{\partial \beta_0} = 0 & \Rightarrow \sum_i \lambda_i y_i = 0, \\ & \lambda_i \geq 0, \\ & \lambda_i y_i (\mathbf{x}_i \beta + \beta_0) - 1 \geq 0, \\ & \lambda_i \{\lambda_i y_i (\mathbf{x}_i \beta + \beta_0) - 1\} = 0. \end{cases}$$

One can find the dual problem which essentially has the same solution as the main problem,

$$\begin{cases} \max_{\lambda_i} \sum_i \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j (\mathbf{x}_i . \mathbf{x}_j) \\ \text{subject to: } \sum_i \lambda_i y_i = 0, \ \lambda_i \geq 0 \end{cases}$$

Now it suffices to solve the dual problem for $\lambda_i > 0$, and find the coefficient vector $\hat{\beta} = \sum_i \lambda_i y_i \mathbf{x}_i$. By picking one example one could find the intercept value β_0. For prediction on new points, we can now do $sign\left(\hat{\beta}.\mathbf{x} + \hat{\beta_0}\right)$.
In some cases we want to compromise a little, increase the training error, but also increase the generalization power(This is also called the error-variance compromise). So we impose slack variables $\xi \geq 0$ which imposes more flexibility on the separation margins,

$$\begin{cases} \beta.\mathbf{x}_i + \beta_0 \geq +1 - \xi_i, & \text{if } y_i = +1 \\ \beta.\mathbf{x}_i + \beta_0 \leq -1 + \xi_i, & \text{if } y_i = -1. \end{cases}$$

Similar to eq. 9.1, one could solve the above program. By changing the values λ_i, one can tune the algorithm to get a better regression ability.
Now let's assume that we want to do some the similar classification on a non-linear space. The idea is to project input data, into higher dimension *feature* space,

$$\Phi : \mathscr{X} \to \mathscr{F}, \ \Phi(\mathbf{x}) = (\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \ldots).$$

So we replace the variable "$\mathbf{x}$", with the new high-dimension variable "$\Phi(\mathbf{x})$". Note that all the above formulation stated, remain the same by the replacement. Now we define notion of *kernel* which appears in different occasions, and gets a more practical interpretations. We define a *kernel* as $k(\mathbf{x}_i,\mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j)\rangle$. To get more intuition into *kernel*s and convince ourselves about necessity of this definition let's go back and see the formulations based on new feature space, $\Phi(\mathbf{x})$. The dual formulation and the prediction formulations are,

$$\begin{cases} \max_{\lambda_i} \sum_i \lambda_i - \frac{1}{2}\sum_{i,j}\lambda_i\lambda_j y_i y_j \overbrace{(\Phi(\mathbf{x}_i).\Phi(\mathbf{x}_j))}^{k(\mathbf{x}_i,\mathbf{x}_j)}, \\ \text{subject to: } \sum_i \lambda_i y_i = 0,\ \lambda_i \geq 0, \\ \text{prediction: } sign\left(\hat{\beta}.\Phi(\mathbf{x}) + \hat{\beta}_0\right) = sign\left(\sum_i \lambda_i y_i \underbrace{(\Phi(\mathbf{x}_i).\Phi(\mathbf{x}))}_{k(\mathbf{x}_i,\mathbf{x})} + \hat{\beta}_0\right). \end{cases}$$

The above formulation of problem, shows how in new formulation the inner product of variables appear in conjunction with each other which we call it *kernel*. Now we can interpret the prediction, as linear combination of kernels, defined by a subset of input data.

9.3 Problems

9.3.1 Text kernels

In this question we will develop a learning algorithm that will take as input a URL and classify it according to whether it is relevant to the topic "Machine Learning" or not. The classifier will only depend on the string of the URL, and not on the web page itself.

In the following we will develop a kernel that will be used to learning how to map a URL string to "relevant" and "irrelevant".

We are given a collection of m URLs $u_1, u_2, \ldots, u_m$. Each URL consists of characters taken from a vocabulary V of n characters $c_1, ..., c_n$. We can assume that V includes *all* ASCII characters.

The *basic* feature vector for each URL u is $F(u)$. $F(u)$ is a binary vector, $F(u) \in \{0,1\}^n$, where the jth component in $F(u)$ indicates whether the character c_j appears in URL u ($F(u)[j] = 1$) or not ($F(u)[j] = 0$). For example, for the URL u = *www.cnn.com*, the set of active features in $F(u)$ is A = *{**w**, **c**, **n**, **.**, **o**, **m**}*, i.e. components in $F(u)$ that correspond to the indices of the characters of A will be 1, all others will be 0.

Each u is also labeled as relevant ($l = 1$) or irrelevant ($l = 0$).

(a) The presence of some *set* of characters can be indicative of "machine learning", e.g. *ml, sv*. So in addition to the basic features in $F(u)$, we want to include features that indicate if a *different* pair of characters c_i and c_j appear anywhere in the URL u. Let us call this new feature space $\phi(u)$.

1. What is the total number of features in the expanded feature space of $\phi(u)$ (as a function of the vocabulary size n)?
   Answer: Including pairs in addition to one-character features will increase the features to $n + \binom{n}{2}$.
2. Assume a URL u = *www.a.sg*
   Write down the active features in the expanded feature vector $\phi(u)$.
   Answer: The new features would be $B = \{w, ., a, s, g\} \cup \{w., .a, .s, a., sg\}$.
3. For a URL u of length s, where all the s characters are different, what is the number of active features in $\phi(u)$ (as a function of s)?
   Answer: $s + \binom{s}{2}$.

4. We want to use the Kernel Perceptron algorithm to learn the function above. Design a kernel $K(u_1,u_2)$ that allows us to compute the value of the dot product $\phi(u_1)\phi(u_2)$ in time linear in n by directly computing it in the $F(u)$ space without expanding to the $\phi(u)$ space (assume that length of u_1 and u_2 is at most n). Give the formula for $K(u_1,u_2)$ and explain why it is true.
   Answer: For each feature in $\phi(u_1)$ we need to check if it is present in the other kernel $\phi(u_2)$ or not. We can do these elementwise comparisons in linear time. The final answer for $K(u_1,u_2)$ is $\#\{\forall i, \phi(u_1)[i] = \phi(u_2)[i]\}$.

(b) Now we are going to define a new feature space $\psi(u)$. $\psi(u)$ will include all features from $F(u)$ and also include features that indicate whether a pair of characters appear consecutively in URL u. For example, for the URL u = *www.cnn.com*, the set of active features in $\psi(u)$ will be A = ***{w, c, n, ., o, m, ww, w., .c, cn, nn, n. , co, om}***.

1. What is the size of the expanded feature space $\psi(u)$ (as a function of the vocabulary size n)?
   Answer: $n + \binom{n}{2}$.
2. For a URL u of length s where all the s characters are different, what is the number of active features of $\psi(u)$ (as a function of s)?
   Answer: $s + s - 1$
3. Write a kernel $K(u_1,u_2)$ that can compute the dot product $\psi(u_1)\psi(u_2)$ in time linear in n (assume that length of u_1 and u_2 is at most n).
   Answer: Just compare all the features one by one in linear time(similar to 2.a.4).

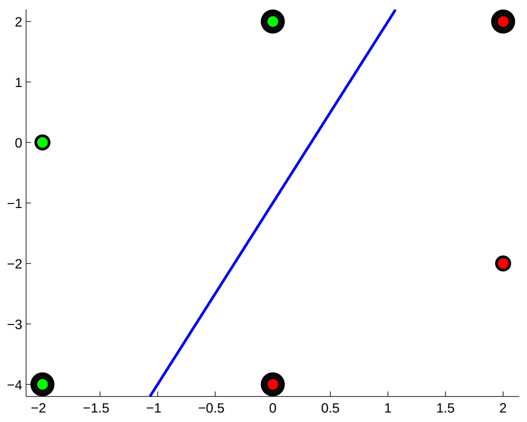


Figure 9.2: Training data, the linear separator and support vectors.

9.3.2 SVM problem

We are given the following set of training examples $D = \{(x_1^{(i)}, x_2^{(i)}, y^{(i)})\}, i = 1, \dots, m$, where $x_j^{(i)}$ are integer-valued features, and $y^{(i)}$ are binary labels.

x_1	x_2	y
-2	-4	+
-2	0	+
0	2	+
2	2	-
2	-2	-
0	-4	-

Our objective is to learn a hyperplane $w_1x_1 + w_2x_2 + b = 0$ using the hard-SVM objective:

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\left(w_1^2 + w_2^2\right) \\ &\text{subject to} && y^{(i)}\left(w_1x_1^{(i)} + w_2x_2^{(i)} + b\right) \geq 1,\ i = 1, \dots, m. \end{aligned}$$

You will need to create a plot to finish the following questions.

1. Finding the hard-SVM hyperplane:
   1. Plot the training examples, and indicate the support vectors.
      Answer: Training data, the linear separator and support vectors are shown in Fig. 9.2.
   2. Draw the hyperplane produced by the hard-SVM..
      Answer: See the Fig. 9.2.
   3. Find the values of $w_1, w_2, b \in \mathbb{R}$ that optimize the hard-SVM objective.
      Answer: See Fig. 9.2. Based on the support vectors we the approximate values for the linear separator are:
      $$w_1 = \frac{1}{3},\ w_2 = -1,\ b = \frac{1}{3}$$
2. Experimental evaluation:

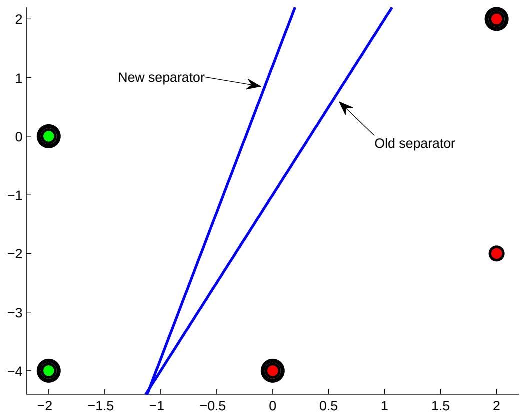


Figure 9.3: New separator when we delete $(0,2,+)$.

1. Provide the classification rule used to classify an example with features x_1, x_2, using the hyperplane produced by hard-SVM.
   Answer: We just need to use the separator line:

$$f(\mathbf{x}) = \begin{cases} +1, & w_1x_1 + w_2x_2 + b \geq 0 \\ -1, & w_1x_1 + w_2x_2 + b < 0 \end{cases}$$

2. What will the error of your classifier be on the training examples D (expressed as the fraction of training examples misclassified)?
   Answer:

$$Error = \frac{|\{\forall(\mathbf{x}_i, y_i) \in D, f(\mathbf{x}_i) \neq y_i\}|}{|D|}$$

3. Draw the hyperplane that will be produced by hard-SVM when you use all training examples except $a = (0,2,+)$. Using this hyperplane, will you classify a correctly?
   Answer: The new separator is shown in Fig. 9.3. Based on the relative distance of $(2,2,+)$ and $(0,0,-)$ from $(0,2,+)$ the new separator will mistake in when making decision for $(0,2,+)$.
4. Draw the hyperplane that will be produced by hard-SVM when you use all training examples except $b = (2,2,-)$. Using this hyperplane, will you classify b correctly?
   Answer: In this case the separator doesn't change. Because if we change it the margin (which is determined by $(-2,-4,+)$ and $(0,-4,-)$) will decrease.
5. What will be the average error if you use 6-fold cross validation on the training set D?
   Answer: In 6-fold cross-validation we divide data into 6 groups and use 5 for training and 1 remaining for testing. Using the previous questions, we can see that among 6 folds, it will make 2 mistakes. Thus, the cross-validation error is $\frac{2}{6}$.

3. Soft-SVM formulation:
   1. Write the soft-SVM objective below. Circle either min or max.
      Answer: For the soft-margin formulation we need the set of slack variables ξ_i for each training set $(\mathbf{x}_i, y_i)$. Thus the new modified constraints become like this:

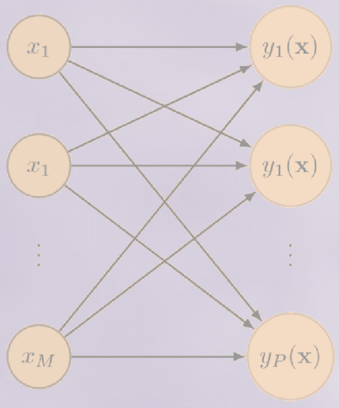
$$y_i.(\mathbf{x}_i.\mathbf{w} + b) \geq 1 - \xi_i, \quad 1 \leq i \leq 6$$

The new objective function becomes as follows(Note that b^2 in the objective function of the question is missing.):

$$\min_{\mathbf{w},b,\xi} \left\{ w_1^2 + w_2^2 + b^2 + C \sum_{i=1}^{6} \xi_i \right\}$$

2. For what range of positive C values, will the hyperplane produced by this soft-SVM objective be most similar to the hyperplane produced by hard-SVM. Circle one of the following.
Answer: When we set $C = \infty$, soft-SVM is equivalent to not having all slack variable be zero; thus for very large values of C, it is the most similar to the hard-SVM.

very small moderate very large



10 — Hidden Markov Models

10.1 Introduction

[TBW]

10.2 Basic Hidden Markov Models

Hidden Markov Models (HMM) are created to model a sequence of events, based on hidden observations. In the demonstrated model in Fig. 11.3, let's assume that $\mathbf{X} = (X_1, \ldots, X_n)$ is the set of observations, and $\mathbf{Z} = (Z_1, \ldots, Z_n)$ is the set of latent states. Each state X_i could take on m_x possible value, and each Z_i could take on m_z possible values. The matrix $A_{m_z \times m_z}$ characterizes the transition probabilities for Z variables. Similarly the matrix $B_{m_z \times m_x}$ denote the probability of observing X_t from Z_t. So we are assuming the Markovian transitions from a hidden-state to a next hidden-states.

In this sequential model, we are in interested in the following probabilities:

- Inference: Given the sequence $(\mathbf{x}, \mathbf{Z})$, what is the probability of $p(\mathbf{x}|\mathbf{Z})$.
- Learning: Given the sequence $\mathbf{x}$ what is the sequence $\mathbf{Z}$ that maximizes $p(\mathbf{Z}|\mathbf{x})$.

Now the question is given a sequence of training data, how can we find the most likely model which has generated the given sequence? We should assume an initial distribution for Z_1 by $w_{m_z \times 1}$. We can write the log-likelihood by

$$\mathscr{L} = \log w(Z_1) + \sum_{t=1}^{n-1} \log A\left(Z_t, Z_{t+1}\right) + \sum_{t=1}^{n} \log B\left(Z_t, X_t\right)$$

We shall train the model using EM algorithm, which also known as Baum-Welch. We define

$$\gamma_t(i,j) = p_{\theta_0}\left(Z_t = i, Z_{t+1} = j | \mathbf{x}\right),\ \gamma_t\left(i\right) = p_{\theta_0}\left(Z_t = i | \mathbf{x}\right) = \sum_j \gamma_t(i,j)$$

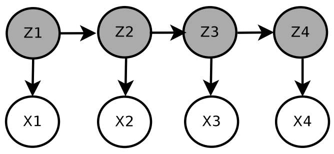


Figure 10.1: Structure of a Hidden Markov Model.

E-Step: In terms of the above definitions,

$$\mathbb{E}_{\mathbf{Z}|\mathbf{x},\theta} \log p(\mathbf{Z},\mathbf{x}|\theta) = \sum_{i=1}^{m_z} \gamma_1(i) \log w(i) + \sum_{i,j=1}^{m_z} \sum_{t=1}^{n-1} \gamma_t(i,j) \log A(i,j) + \sum_{i=1}^{m_z} \sum_{t=1}^{n} \gamma_t(i) \log B(i,x_t)$$

M-Step: and using derivatives of the the above term we have the following estimated values:

$$w^*(i) = \gamma_1(i),\ i = 1,\dots,m_z;$$

$$A^*(i,j) = \frac{\sum_{t=1}^{n-1} \gamma_t(i,j)}{\sum_{j'} \sum_{t=1}^{n-1} \gamma_t(i,j')},\ i,j = 1,\dots,m_z;$$

$$B^*(i,l) = \frac{\sum_{t=1;x_t=l} \gamma_t(i)}{\sum_t \gamma_t(i)},\ i = 1,\dots,m_z,\ l = 1,\dots,m_x;$$

Forward-Backward probabilities: Define forward $\alpha_{t+1}(i)$ and backward $\beta_{t-1}(i)$ probabilities as following:

$$\alpha_{t+1}(i) = p_\theta(x_1,\dots,x_{t+1},Z_{t+1} = i),\ \ \beta_{t-1}(i) = p_\theta(x_t,\dots,x_n|Z_{t-1} = i)$$

We can show that the forward probability $\alpha_{t+1}(i)$ could be expressed based of the previous ones:

$$\begin{aligned}\alpha_{t+1}(i) &= p_\theta(x_1,\dots,x_{t+1},Z_{t+1} = i) && (10.1a)\\ &= \sum_j p_\theta(x_1,\dots,x_{t+1},Z_t = j,Z_{t+1} = i) && (10.1b)\\ &= \sum_j \alpha_t(j)A(j,i)B(i,x_{t+1}) && (10.1c)\end{aligned}$$

Similarly the backward probability could be written based on the future ones:

$$\begin{aligned}\beta_{t-1}(i) &= p_\theta(x_t,\dots,x_n|Z_{t-1} = i) && (10.2a)\\ &= \sum_j p_\theta(x_t,\dots,x_n,Z_t = j,Z_{t-1} = i) && (10.2b)\\ &= \sum_j A(i,j)B(j,x_t)\beta_t(j) && (10.2c)\end{aligned}$$

Now we can define the joint probabilities in terms of $\alpha(.)$ and $\beta(.)$.

$$\begin{aligned}p(Z_t = i,x_1,\dots,x_n) &= p(x_1,\dots,x_n|Z_t = i).p(Z_t = i) && (10.3a)\\ &= p(x_1,\dots,x_t,Z_t = i).p(x_{t+1},\dots,x_n|Z_t = i) && (10.3b)\\ &= \alpha_t(i).\beta_t(i) && (10.3c)\end{aligned}$$

And also we can express the whole likelihood in terms of forward probabilities:

$$p(\mathbf{x}|\theta) = \sum_i p_\theta(x_1,\dots,x_n,Z_n = i) = \sum_i \alpha_n(i)$$

The computation of all probabilities in terms of forward/backward probabilities makes everything so simple. We just need to calculate them in two swipes. It turns out that forward-backward algorithms is an special case of *belief-propagation* algorithm in Graphical Models. Now the question is given a model, how can we make inference about the hidden states. The inference about the hidden variables depend on the definition of the optimality for the inference. One criterion is to choose the hidden values that are individually most likely, that is:

$$\hat{Z}_t^* = \arg\max_i p(Z_t = i|\mathbf{x}, \theta) = \arg\max_i \gamma_t(i)$$

But the problem is that considering each observation individually is not reasonable, and one needs to choose the most-likely sequence:

$$\hat{\mathbf{Z}}^* = \arg \max_{i_1,\dots,i_n} p(Z_1 = i_1, \dots, Z_n = i_n|\mathbf{x}, \theta)$$

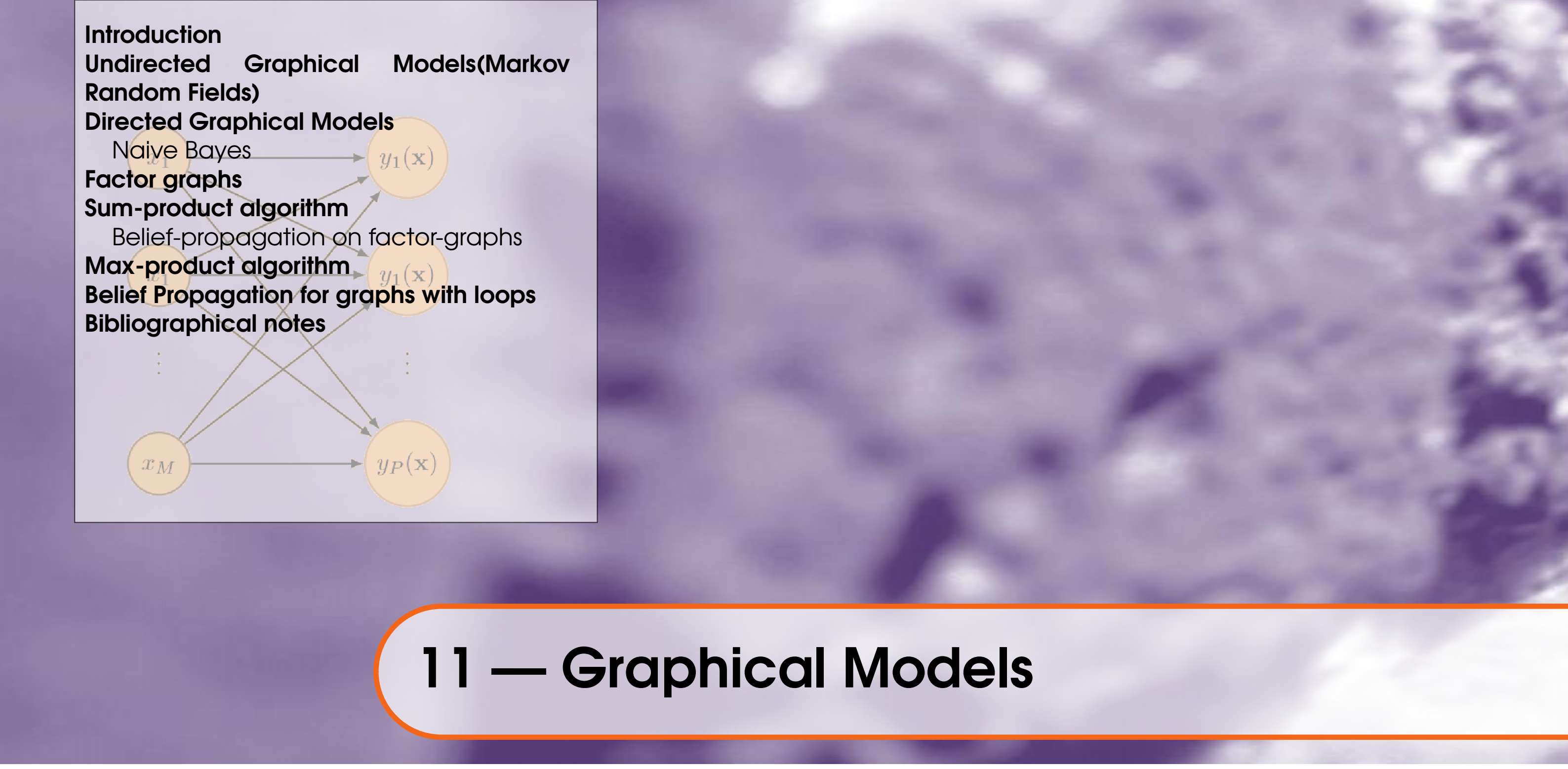
This solution could be found using Viterbi algorithm. Assume that we have a trained HMM and we want to find the most probable latent path, given some observations. The highest probability of the sequence of hidden states given first t observations is:

$$\delta_t(i) = \max_{j1,\dots,j_{t-1}} p(Z_1 = j_1, \dots, Z_{t-1} = j_{t-1}, Z_t = i, \mathbf{x}_{1:t}|\theta)$$

Rewriting the optimality criterion it a recursive form :

$$\delta_{t+1}(i) = \left[\max_j \delta_t(j) A(j,i)\right] .B(i, x_{t+1})$$

Similar implementation to forward/backward algorithm, will give us the optimal latent sequence. This turns out that Viterbi is an special case of *max-product* algorithm in graphical models.



11 — Graphical Models

11.1 Introduction

[TBW]

11.2 Undirected Graphical Models(Markov Random Fields)

Undirected graph $G = (V, E)$ with vertices on the set of variables $X = (X_1, \ldots, X_n)$. In this notation $n = |V|$ and $m = |E|$.

The graph shows the connection (interaction/dependence) between the variables in the model. Thus it is more accurate to model the set of variables connected to each other jointly. The most *accurate* modelling is defining a joint distribution for any variables in a connected graph. But such modelling strategy will create very big and cumbersome distribution which are hard to train, and do inference with. Thus we prefer to do approximations. One of such approximations is the *Markov property* in writing the joint distribution between the variables in the graph.

Definition 11.1 — Markov property in undirected graphical models. Let's say we have a graph $G = (V, E)$ in which the vertices represent the set of variables, and the edges represent the connection between the variables. The variables have the *Markov property* if for any two set of variables X_A and X_B (corresponding to the set of variables A and B respectively), if there is a set of variables X_S and S *separates* A and B, the variables X_A and X_B are independent, conditioned on X_S. Here "S *separates* A and B" means that, any path from any vertex in A to any vertex in B, needs to use at least one of the vertices in S. The mathematical notation for this definition is the following,

$$X_A \perp\!\!\!\perp X_B | X_S$$

Based on the above definition we can create the following two corollaries as a result of Makov property assumption.

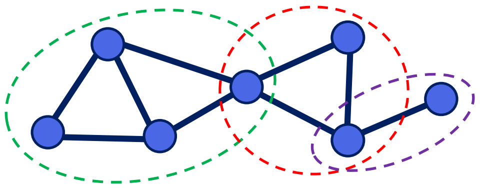


Figure 11.1: An undirected graphical models

Corollary 11.1 **Global Markov property:** For any two vertices which are not neighbours, given the other vertices they are independent:

$$X_a \perp\!\!\!\perp X_b | X_{V \backslash \{a,b\}} \Leftrightarrow (a,b) \notin E$$

Local Markov property: Any vertex is independent of all vertices, but its neighbours and itself, conditioned on all of its neighbours.

$$X_a \perp\!\!\!\perp X_{V \backslash \{a, \text{ne}(a)\}} | X_{\text{ne}(a)}$$

Note that the *local* Markov property is stronger that the *global* one.

To define proper factorization we need to be more careful. We need to define a factorization, as simple as possible, on the given input given graph, with respect to an input graph with its given edge connections, and obey the Markov independence property. Let's define some more mathematical definitions for graph properties. A *clique* C is a set of vertices, which are all connected to each other. For example in the Figure **??** there are three cliques are depicted (red, purple and green). The easiest way to define the factorization of distributions is to use cliques,

$$p(\mathbf{X}) = \prod_{C \in \text{clique}(G)} p(\mathbf{X}_C). \tag{11.1}$$

This is a valid factorization since it has the *local* Markov property (proof?).

Definition 11.2 — Gibbs Random Field. When the probability factors defined for each of the factors of the Markov random field are strictly positive, we call it Gibbs random field.

Theorem 11.1 — HammersleyClifford theorem. A positive probability distribution with continuous density satisfies the pairwise Markov property with respect to an undirected graph G if and only if it factorizes according to G.

In other words, Hammersley-Clifford is saying that for any strictly positive distribution $p(.)$, the Markov property is equivalent to the factorization in Equation 11.1.

■ **Example 11.1 — Discrete Markov Random Field.** Let $G = (E, V)$ be an undirected graph. Each vertex is associated with a random variable $x_s \in \{0, 1, \ldots, d-1\}$ Assume the following exponential density,

$$p(\mathbf{x}; \theta) \propto \exp \left\{ \sum_{v \in V} \theta_v(x_v) + \sum_{(v,u) \in E} \theta_{vu}(x_v, x_u) \right\},$$

where the set of factors used in the definition of the above function are,

$$\begin{cases} \theta_v(x_v) = \sum_{j=0}^{d-1} \alpha_{v;j}\mathbf{I}_j(x_v) \\ \theta_{vu}(x_v, x_u) = \sum_{j=0}^{d-1}\sum_{k=0}^{d-1} \alpha_{vu;jk}\mathbf{I}_j(x_v)\mathbf{I}_k(x_u) \end{cases}$$

where $\mathbf{I}_.(.)$ is the indicator function,

$$\mathbf{I}_j(x_v) = \begin{cases} 1 & \text{if } x_v = j \\ 0 & \text{otherwise} \end{cases}$$

and the parameters of the model are $\theta = \left\{\alpha_{v;j} \in \mathbb{R}, v \in V, j \in \mathscr{X}\right\} \cup \left\{\alpha_{vu;jk} \in \mathbb{R}, (v,u) \in E, (j,k) \in \mathscr{X} \times \mathscr{X}\right\}$. The cumulant generating function (log normalization function) is,

$$A(\theta) = \log \sum_{x\in\mathscr{X}} \exp\left\{ \sum_{v\in V} \theta_v(x_v) + \sum_{(v,u)\in E} \theta_{vu}(x_v, x_u) \right\},$$

■

■ **Example 11.2 — Ising Model.** Ising Model is an special case of Discrete Markov Random Field, when the variable can get only two values $x_s \in \{0,1\}$ $(d = 2)$. The probability distribution can be shown in a more simpler form,

$$p(\mathbf{x};\theta) \propto \exp\left\{ \sum_{v\in V} \theta_v x_v + \sum_{(v,u)\in E} \theta_{vu} x_v x_u \right\},$$

and the parameters of the model are $\theta = \{\theta_v \in \mathbb{R}, v \in V\} \cup \{\theta_{vu} \in \mathbb{R}, (v,u) \in E\}$. In this exponential representation the vector is $\phi(\mathbf{x}) = [\ldots x_v \ldots x_{vu} \ldots]^\top$, with length $|V| + |E|$. This exponential family is *regular* with $\Omega = \mathbb{R}^{|V|+|E|}$, and a *minimal* representation(if you don't know what each of these properties mean, refer to Wiki).

■

11.3 Directed Graphical Models

Directed graphical models are directed acyclic graphs (DAG), which accept factorization of distributions based on child-parent order,

$$p(\mathbf{X}) = \prod_{u\in V} p(u|\text{pr}(u)),$$

where $\text{pr}(u)$ is the set of parent vertices for u. An example directed graphical model is depicted in Figure **??**, in which is the set of parents for the orange vertex are denoted by a green oval.

■ **Example 11.3 — HMM.** An example for directed graphical models is Hidden Markov Models (HMM). The joint distribution for the observations and the latent variables shown in Figure 11.3 in an HMM is as following:

$$p(\mathbf{X},\mathbf{Z}) = p(Z_1).p(Z_1|Z_1).p(Z_2|Z_1).p(X_2|Z_2)\ldots. = p(X_1|Z_1)p(Z_1)\prod_{i=2}^{n} p(X_i|Z_i)p(Z_i|Z_{i-1}).$$

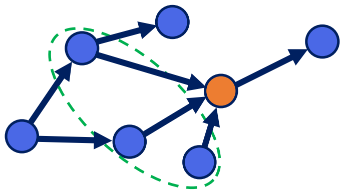


Figure 11.2: A directed graphical model; the parents of the node in orange is depicted in a green oval.

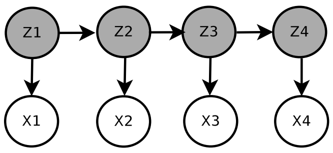


Figure 11.3: Structure of a Hidden Markov Model.

It should be clear that the above decomposition of probability distributions for an HMM exactly follows the definition of a directed graphical model. ■

11.3.1 Naive Bayes

In *Naive Bayes Classifier*, we assume that independence of input features, given the class, that is, we can write their joint distribution in the following way:

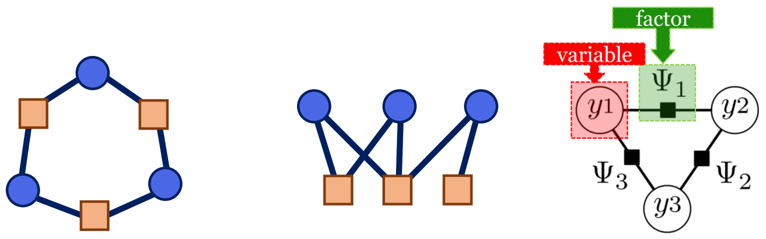
$$f(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n | G = g) = \prod_{i=1}^{n} f(\mathbf{X}_i | G = g)$$

Note that in the above formulation, $\mathbf{X}_i$ refer to each of input features, each of which could be of several dimension(and should not be confused with dimensions of one single feature, though in some cases they can be tough as the same thing). By applying this assumption in equation 4.2, we have:

$$\begin{aligned}\log \frac{\Pr(G = k | X = x)}{\Pr(G = l | X = x)} &= \log \frac{\pi_l f(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n | G = l)}{\pi_k f(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n | G = k)} \\ &= \log \frac{\pi_l \prod_{i=1}^{n} f(\mathbf{X}_i | G = l)}{\pi_k \prod_{i=1}^{n} f(\mathbf{X}_i | G = k)} \\ &= \log \frac{\pi_l}{\pi_k} + \sum_{i=1}^{n} \log \frac{f(\mathbf{X}_i | G = l)}{f(\mathbf{X}_i | G = k)}\end{aligned}$$

11.4 Factor graphs

Factor graphs, unifies the directed and undirected representation of graphical models Frey (2002). As it is shown in Figure 11.4, there are two types of nodes, factor nodes, and variable



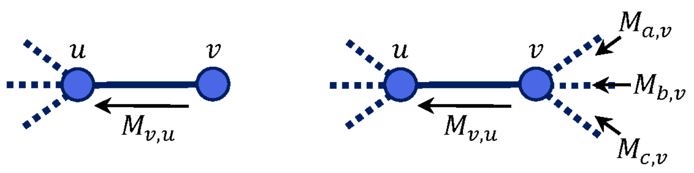


Figure 11.5: The elimination of a leaf node.

nodes. In addition to vertices that represent random variables (as it was the case in Directed and Undirected Graphical Models), we introduce rectangular vertices to the representation. Two examples are depicted in Figure 11.4. The nodes stand for random variables, and the rectangles are the joint functions among the variables which they are connected to. These representation can give more accurate decomposition of probability factors for the variables. The fact that factor graphs combine undirected and directed graphical models is because, the function/distribution in the factors could be anything, e.g. the joint distribution or a conditional distribution. To have a general representation we show each function/distribution defined on each factor by $f(\mathbf{x})$ where $\mathbf{x}$ is the set of variables which are connected to that factor. For example for the factor graph in Figure 11.4, the expansion of the distribution is as following,

$$p(x_1,x_2,x_3) \propto f(x_1,x_2)f(x_2,x_3)f(x_1,x_3)$$

11.5 Sum-product algorithm

Let's say we have a acyclic graph and we want to calculate the marginals. Let's say we have the following factorization on a an arbitrary acyclic graph,

$$p(\mathbf{x}) = \frac{1}{\mathscr{Z}} \prod_{s\in V} \psi(\mathbf{x}_s) \prod_{(u,v)\in E} \psi(\mathbf{x}_u,\mathbf{x}_v) \tag{11.2}$$

Let's say we are given a graph and we want to marginalize over a leaf variable $\mathbf{x}_v$ (figure 11.5(left)),

$$p(\mathbf{x}_{\setminus v}) \propto \sum_{\mathbf{x}_v} \prod_{s\in V} \psi(\mathbf{x}_s) \prod_{(a,b)\in E} \psi(\mathbf{x}_a,\mathbf{x}_b),$$

which can be simplified as,

$$p(\mathbf{x}_{\setminus v}) \propto \left(\prod_{s\in V\setminus\{v\}} \psi(\mathbf{x}_s) \prod_{(a,b)\in E,\ v\neq t} \psi(\mathbf{x}_a,\mathbf{x}_b) \right) \cdot \sum_{\mathbf{x}_v} \left(\psi(\mathbf{x}_v) \prod_{(a,v)\in E} \psi(\mathbf{x}_a,\mathbf{x}_v) \right).$$

Like the case of Figure 11.5 if we assume that v (the leaf) has only one parent u we can simplify the equation more,

$$p(\mathbf{x}_{\backslash v}) \propto \left(\prod_{s \in V \backslash \{v\}} \psi(\mathbf{x}_s) \prod_{(a,b) \in E,\ v \neq t} \psi(\mathbf{x}_a, \mathbf{x}_b) \right) . \sum_{\mathbf{x}_v} \psi(\mathbf{x}_v) \psi(\mathbf{x}_u, \mathbf{x}_v). \tag{11.3}$$

Thus the marginalization over the leaf variables is equivalent to marginalization over the factors of the leaf variables, and the mutual factors with its parents, and multiplying the result to the rest of the product. This idea makes the problem of inference on acyclic graphs much easier. Instead of global summations (integrations) we only need local integrations by carefully ordering the variables. Now we make the notation more concrete by defining *message*. We define $M_{v,u}(\mathbf{x}_u)$ as the message from v(a leaf) to u, which is the marginalization over the sender's and mutual factors, and is only a function of the recipient.

$$M_{v \to u}(\mathbf{x}_u) = \sum_{\mathbf{x}_v} \psi(\mathbf{x}_v) \psi(\mathbf{x}_u, \mathbf{x}_v).$$

So, we can say that, when marginalization of leaf variables, we only need to keep the messages and through anything else away.

Let's say we have a more general case, as in the Figure 11.5(right) where we are marginalizing over a set of variables, including v which is not a leaf. For simplicity we can assume that the others are three leaf nodes, a, b, c. Since we want to marginalize over $\{v, a, b, c\}$, we can start by calculating the messages from the leaves $M_{a,v}(\mathbf{x}_v), M_{b,v}(\mathbf{x}_v)$ and $M_{c,v}(\mathbf{x}_v)$ and multiplying them into the joint distribution of the remaining variables. Now we only need to marginalize over v:

$$p(\mathbf{x}_{\backslash v}) \propto \left(\prod_{s \in V \backslash \{v\}} \psi(\mathbf{x}_s) \prod_{(a,b) \in E,\ v \neq t} \psi(\mathbf{x}_a, \mathbf{x}_b) \right) . \sum_{\mathbf{x}_v} \psi(\mathbf{x}_v) \psi(\mathbf{x}_u, \mathbf{x}_v) \underbrace{[M_{a \to v}(\mathbf{x}_v) M_{b \to v}(\mathbf{x}_v) M_{c \to v}(\mathbf{x}_v)]}_{\text{Messages from children}}.$$

The only difference between the above equation and Equation 11.3 is the last term which is the *messages from children*. In other words, to marginalize over a non-leaf variable v, we only need to calculate the messages from its children, then sum over,

- The messages from its children (here a, b and c)
- The factor on itself, v
- The joint factor of it with its parent u

Now we can give the the general definition of a message,

$$M_{v \to u}(\mathbf{x}_u) = \sum_{\mathbf{x}_v} \psi(\mathbf{x}_v) \psi(\mathbf{x}_u, \mathbf{x}_v) \prod_{a \in N(v) \backslash \{u\}} M_{a \to v}(\mathbf{x}_v).$$

In the above definition $N(t)$ is the set of all vertices which neighbour v. Based on the above message formula we can calculate the marginals as following,

$$p(\mathbf{x}_s) \propto \psi_t(\mathbf{x}_t) \prod_{t \in N(s)} M_{t \to s}(\mathbf{x}_s)$$

Corollary 11.2 Since any tree is a acyclic graph, the marginals can be calculated in one-time swiping the nodes of the tree. In other words, the sum-product algorithm can give the exact marginal distributions of every node in a tree in $O(n = |V|)$ complexity.

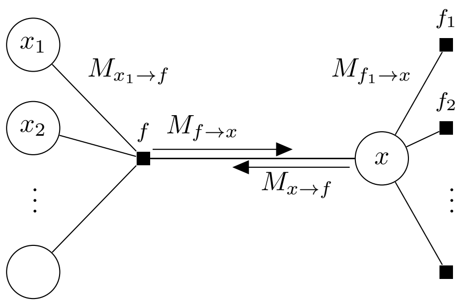


Figure 11.6: Message passing on a factor graph.

11.5.1 Belief-propagation on factor-graphs

It is easy to generalize the belief-propagation scheme mentioned in the previous section to factor-graphs. There is a nice discussion of factor graphs could be found at Kschischang et al. (2001). The messages from variables to factors is,

$$M_{f \to x}(x) = \prod_{f' \in N(x) \setminus \{f\}} M_{f' \to x}(x).$$

And messages from factors to variables,

$$M_{x \to f}(x) = \sum_{\mathbf{x} \setminus x} f(\mathbf{x}) \prod_{x' \in N(f) \setminus \{x\}} M_{x' \to x}(f).$$

Note that in the above notation, $N(f)$ is the set of all *variables* adjacent to factor f, and $N(x)$ is the set of all *factors* adjacent to variable x. The belief propagation for factor graphs is depicted in Figure 11.6.

When the

11.6 Max-product algorithm

Before anything, we show the necessity of finding max-probability for a set of random variables.

■ **Example 11.4 — Inference in graphical models.** Let's consider a general undirected graphical model with set of cliques $\mathscr{C}$,

$$p(\mathbf{x}) = \frac{1}{\mathscr{Z}} \prod_{C \in \mathscr{C}} \psi(\mathbf{x}_C)$$

with a set of observation variables $\mathbf{x}_E$. We want to maximize the posterior probability of variables given some observations O ,

$$\theta = \arg\max_{\theta} p(\mathbf{x} = O)$$

■

The max-product algorithm is so similar to the sum-product algorithm we discussed. In the max-product algorithm we assume that we have a joint distribution of a group of variables, and we want to find its mode with respect to several variables. Let's say we have the same factorization as in Equation 11.2. It it easy to observe that *maximization* has distributive property on factors, like summation in the sum-product algorithm. Thus, if in the sum-product we substitute the summation with maximization, we will end-up with the desired algorithm,

$$M_{v\to u}(\mathbf{x}_u) = \max_{x'_v \in \mathscr{X}_v} \left\{ \psi(\mathbf{x}'_v)\,\psi(\mathbf{x}_u, \mathbf{x}'_v) \prod_{a\in N(v)\setminus\{u\}} M_{a\to v}(\mathbf{x}_v) \right\}.$$

And the *max-marginal* can be found using the following,

$$\tilde{p}(\mathbf{x}_u) \propto \psi_u(\mathbf{x}_u) \prod_{v\in N(u)} M_{v\to u}(\mathbf{x}_u)$$

Corollary 11.3 Since any tree is a acyclic graph, the max-marginals can be calculated in one-time swiping the nodes of the tree. In other words, the max-product algorithm can give the exact max-marginal distributions of every node in a tree in $O(n = |V|)$ complexity.

11.7 Belief Propagation for graphs with loops

In general in a graph with cycles, there might be cliques that create groups of vertices that are all connected to each other. If we find all the cliques in a graph, and group the vertices inside them together we will the *clique tree* of a graph, in which nodes are cliques of the graph. The separator sets are formed by the intersections of cliques adjacent in the graph.

Definition 11.3 — Running intersection property. A graph has the running intersection property if for any two cliques C_1 and C_2 for any unique paths joining them include the intersection nodes $C_1 \cap C_2$.

A clique tree with running intersection property is called a *junction tree*. This property is important because it ensures the probabilistic consistency of the calculations in a clique tree.

Definition 11.4 — Triangulated graph. A graph is triangulated (or chordal, rigid circuit) if each of its cycles of four or more nodes has an edge joining two nodes that are not adjacent in the cycle. Such an edge is called a *chord*.

Proposition 11.1 — ?. A graph has a junction tree if and only if it is triangulated (proof?).

The junction tree for exact inference is as following,

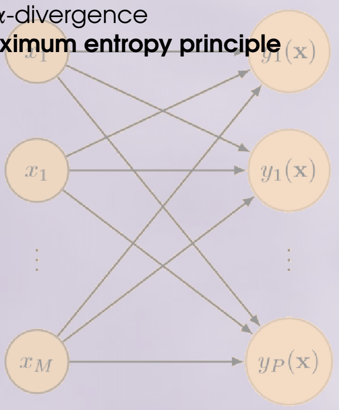
1. Given an undirected graph G form a triangulated graph $\tilde{G}$ by adding edges.
2. Form the clique graph (in which nodes are cliques of the triangulated graph).
3. Extract a junction tree (using a maximum weight spanning tree algorithm on weighted clique graph).
4. Run sum/max-product on the resulting junction tree.

Proposition 11.2 In a triangulated graph, there is an elimination ordering for G that does not lead to any added edges (proof?).

Theorem 11.2 If in a triangulated graph, for any adjacent cliques A and B, we weight all edges of a cliques with $|A \cap B|$, then the junction tree of the clique graph could be find by maximum-weight spanning tree in linear time (proof?).

11.8 Bibliographical notes

In preparation of this notes I have used Martin J. Wainwright's slides. Thanks to `https://github.com/jluttine/tikz-bayesnet`, many graphical models are drawn with this package.



12 — Information theory for learning

12.1 Introduction

[TBW]

12.2 Distribution divergences

12.2.1 Kullback-Leibler divergence

One of the famous distance measure between functions is called Kullback-Leibler divergence or in short, KL-*divergence*. Let's assume we have two functions $f(x)$ and $g(x)$ defined on the same domain $\mathscr{X}$. The distance between these two functions using KL-*divergence* is,

$$\mathrm{KL}(f||g) = \int_{x \in \mathscr{X}} f \log \frac{f}{g} dx$$

it should be easy to see how to use this definition for discrete domains

$$\mathrm{KL}(f||g) = \sum_{x \in \mathscr{X}} f \log \frac{f}{g}$$

one could check that the above measure has the two following properties,

$$\mathrm{KL}(f||g) \geq 0 \tag{12.1a}$$

$$\mathrm{KL}(f||g) = 0 \Longleftrightarrow f(x) = g(x), \forall x \in \mathscr{X} \tag{12.1b}$$

The second property shows that this measure isn't symmetric, i.e. $\mathrm{KL}(p||q) \neq KL(q||p)$; this it might not be an exact *distance* measure. You can check the main paper, Kullback and Leibler (1951) to see more details on KL-*divergence*.

12.2.2 α-divergence

α-divergence or alpha-divergence is a generalization of *KL*-divergence, which is defined as followiing:

$$D_\alpha(p||q) = \frac{\int \alpha p(\mathbf{x}) + (1-\alpha) q(\mathbf{x}) - p^\alpha(\mathbf{x}) q^{1-\alpha}(\mathbf{x}) d\mathbf{x}}{\alpha(1-\alpha)}$$

where $\alpha \in \mathbb{R}$. Some properties of this divergence are,

1. The α-divergence is convex with respect to $p(.)$ and $q(.)$
2. $D_\alpha(p||q) \geq 0$
3. $D_\alpha(p||q) = 0$ iff $p = q$. (This is easy to check from the definition)

For different values of α the above divergence has very interesting properties. Some of these these speciall cases are listed here:

1. $\lim_{\alpha\to 0} \mathrm{D}_\alpha(p||q) = \mathrm{KL}(q||p)$
2. $\lim_{\alpha\to 1} \mathrm{D}_\alpha(p||q) = \mathrm{KL}(p||q)$
3. $\mathrm{D}_{-1}(p||q) = \frac{1}{2}\int \frac{(q(\mathbf{x})-p(\mathbf{x}))^2}{q(\mathbf{x})} d\mathbf{x}$
4. $\mathrm{D}_2(p||q) = \frac{1}{2}\int \frac{(q(\mathbf{x})-p(\mathbf{x}))^2}{p(\mathbf{x})} d\mathbf{x}$
5. $\mathrm{D}_{\frac{1}{2}}(p||q) = 2\int \left(\sqrt{p(\mathbf{x})} - \sqrt{q(\mathbf{x})}\right)^2 d\mathbf{x}$
   It is easy to see that in this case when $\alpha = 0.5$ the divergence is symmetric. This is known as Hellinger distance. The square root of this divergence satisfies the triangle inequality.

12.3 Maximum entropy principle

The exponential distribution also arises naturally from the *maximum entropy principle*. The maximum entropy procedure consists of seeking the probability distribution which maximizes information entropy, subject to the constraints of the information. If we constrain the expected values of the sufficient statistics to be mean of the empirical mean of them under the sampled data, the resulting distribution which maximizes the entropy is the exponential family. To make the statement more accurate, let's express it in terms of formula. Given iid random variables, $\mathbf{x}_1, \ldots, \mathbf{x}_n \sim p^*$, where p^* is the underlying *unknown* distribution. Define a set of functions $\{\phi_i(.) : \mathscr{X} \to \mathbb{R}\}_{i\in\mathscr{I}}$, and consider the empirical mean of the sampled data under these functions,

$$\hat{\mu}_j = \frac{1}{n}\sum_{i=1}^{n} \phi_j(\mathbf{x}_i),\ \forall j \in \mathscr{I}$$

The problem is that we are looking for a distribution, $p(.)$ which has the same set of expected values for the defined functions as their empirical distribution, i.e.

$$\mathbb{E}_p[\phi_j(\mathbf{x})] = \int_{\mathscr{X}} \phi_j(\mathbf{x})p(\mathbf{x})\nu(d\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\phi_j(\mathbf{x}_i) = \hat{\mu}_j,$$

and at the same time maximizes the entropy under this distribution,

$$H(p) = \int_{\mathscr{X}} -p(\mathbf{x})\log p(\mathbf{x})\nu(d\mathbf{x}).$$

This constrained optimization will result in the parametric form of the exponential family of distributions defined here. In other words,

$$\begin{cases} \max_p H(p) = \max_p \int_{\mathscr{X}} -p(\mathbf{x})\log p(\mathbf{x})\nu(d\mathbf{x}), \\ \text{such that, } \mathbb{E}_p[\phi_j(\mathbf{x})] = \hat{\mu}_j \end{cases}$$

The answer to the this constrained optimization problem is an exponential form,

$$\hat{p}(\mathbf{x};\theta) \propto \exp\left\{\sum_{\alpha\in\mathscr{I}} \theta_\alpha \phi_\alpha(\mathbf{x})\right\}$$

■ **Lemma 12.1 — From Maximum Entropy to Exponential Families.** Finding maximum entropy (least informative) distribution with given expectation with respect to a set of functions, $\{\Phi(\mathbf{x})\}_i$, will result in the exponential families,

$$\begin{cases} \max E = \max\{-\int_{\mathscr{X}} p(\mathbf{x}) \log p(\mathbf{x}) d\mathbf{x}\} \\ \mathbb{E}_p[\phi(\mathbf{x})] = \mu \end{cases}$$

is equivalent to,

$$\max\{\langle\mu,\theta\rangle - \mathscr{Z}(\theta)\}$$

where $\mathscr{Z}(\theta) = \int_{\mathscr{X}} \exp(\phi(\mathbf{x}),\theta)\, d\mathbf{x}$

Proof. This can be shown using Lagrange dual

$$\begin{aligned} L(p,\theta,\alpha,\beta) = &-\int_{\mathscr{X}} p(\mathbf{x}) \log p(\mathbf{x}) d\mathbf{x} \\ &+ \alpha^\top \left(1 - \int_{\mathscr{X}} \exp(\phi(\mathbf{x}),\theta)\, d\mathbf{x}\right) \\ &+ \beta^\top \left(\mu - \int_{\mathscr{X}} \phi(\mathbf{x}) p(\mathbf{x}) d\mathbf{x}\right) \end{aligned}$$

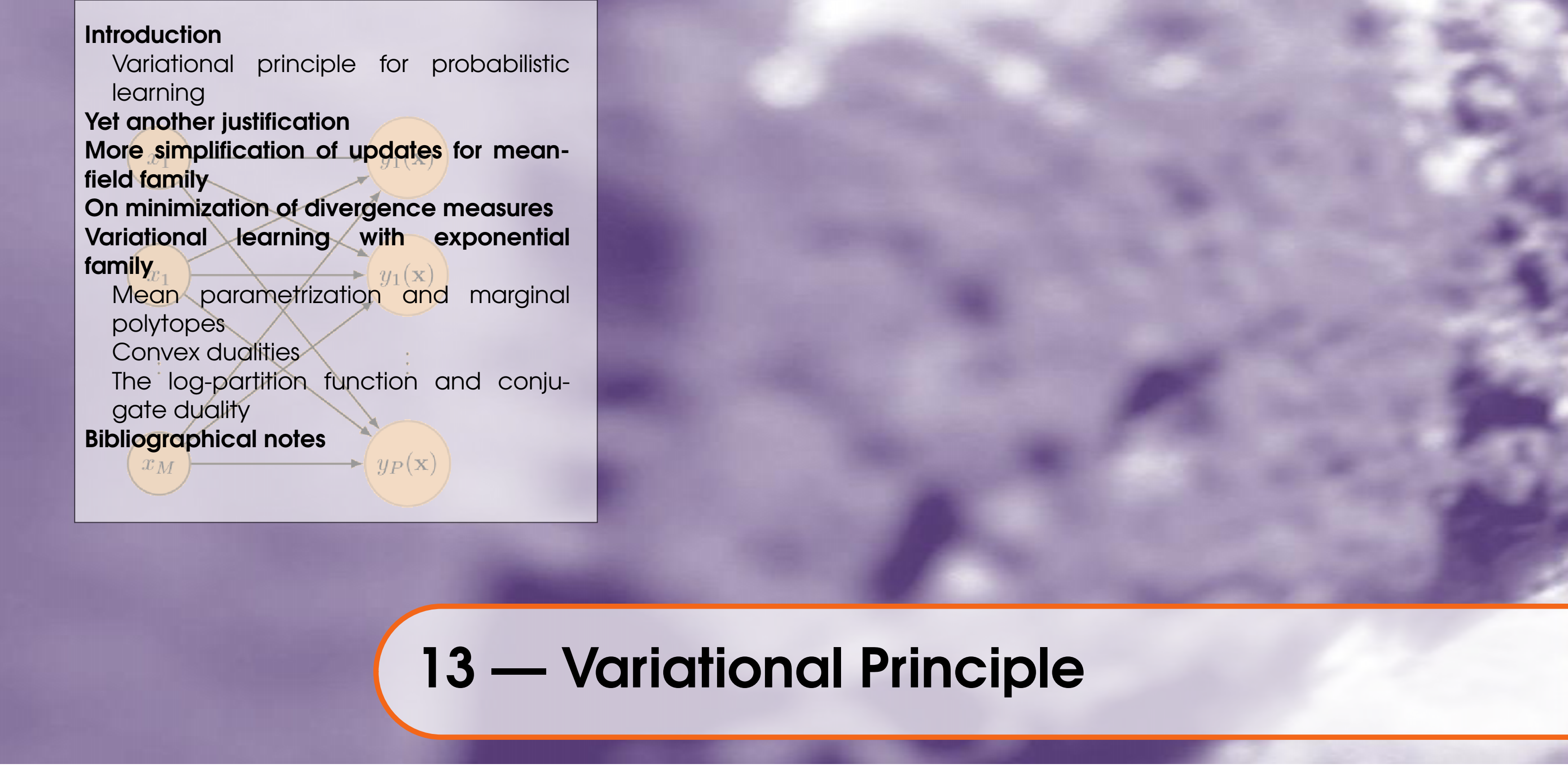
Taking functional derivative with respect to $p(.)$,

$$\frac{\partial L}{\partial p} = -\log p(\mathbf{x}) + \beta + 1 + \theta^\top \phi(\mathbf{x}) = 0$$

$$\Rightarrow p(\mathbf{x}) = \exp\left(\theta^\top \phi(\mathbf{x}) + \beta + 1\right)$$

$$\Rightarrow p(\mathbf{x}) = \frac{1}{\mathscr{Z}(\theta)} \exp\left(\theta^\top \phi(\mathbf{x})\right),\ \mathscr{Z}(\theta) = \int_{\mathscr{X}} \exp(\phi(\mathbf{x}),\theta)\, d\mathbf{x}$$

■



13 — Variational Principle

13.1 Introduction

The name “Variational" is a general name to call a very broad range of optimization-based formulation of problems, which are mostly inspired from “calculus of variations". Thus one should NOT look at this method, as a black-box algorithm; instead it has includes a very useful techniques for simplifying very broad range of problems into tractable optimization problems.

■ **Example 13.1 — Variational representation for solving linear systems. Problem:** Let’s say we are given a vector $\mathbf{y} \in \mathbb{R}^n$ and a positive-definite matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, and we want to solve the given linear system $\mathbf{Ax} = \mathbf{y}$.
Direct solution: The direct solution could be found from the matrix inversion:

$$\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{y}$$

Equivalent variational solution: Assume the following cost function:

$$J_{\mathbf{y}}(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Ax} - \mathbf{y}^\top \mathbf{x}.$$

The notation $J_{\mathbf{y}}(\mathbf{u})$ means that this cost function is a parametric form of the observation $\mathbf{y}$ and the variable $\mathbf{u}$. We can show that this cost function is strictly convex and the minimum fix-point equals to $\mathbf{x}^* = \arg\min_{\mathbf{x}} J_{\mathbf{y}}(\mathbf{u}) = \mathbf{A}^{-1}\mathbf{y}$.
The above two solutions (direct and variational) are both equivalent. This example shows how the prevalent problem of matrix inverse and linear systems can be posed as an optimization problem. ■

13.1.1 Variational principle for probabilistic learning

Variational is a very useful approach to solving problems with intractable posterior distribution via some approximations. The idea is to replace the *intractable* posterior with a *tractable* approximation. Here I am going to symbolically introduce the variational principle for learning probabilistic models. First assume the simple graphical model in Figure 13.1, in which we assume $\mathbf{X} \in \mathbb{R}^n$ and $\mathbf{Z} \in \mathbb{R}^m$ which are two random variables. Our observations is $\mathbf{X}$, and using that, we want to learn and infer about the latent variable $\mathbf{Z}$. Let’s also assume that we have a set of parameters θ. Note that, this simple graphical model is just a symbolic representation of a our main big graphical, and in practice we might have a set of observations and latent variables. In probabilistic way we can show this inference as posterior on latent variables, conditioned on observations $p(\mathbf{Z}|\mathbf{X},\theta)$. We assume that the main model is complicated that this posterior is intractable:

$$p(\mathbf{X},\mathbf{Z}|\theta) = p(\mathbf{Z}|\mathbf{X},\theta)p(\mathbf{X}|\theta). \tag{13.1}$$

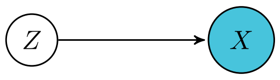


Figure 13.1: Symbolic connection between observation and latent variables. The blue variable is observation.

In variational learning we are going to approximate the real posterior distribution with a another distribution. Generally one can get an approximation to another distribution, by assuming some independence assumptions which simplify it into multiplication of several simpler distributions which are easier to work with.For the sake of emphasis on *approximation*, it is common to denote this distribution with $q(.)$ instead of $p(.)$:

$$q(\mathbf{Z}) = \prod_{i=1}^{m} q(Z_i) \tag{13.2}$$

There is not general rule for decomposition of variables into disjoint distributions; for any problem one should consider the interaction between variables and possibility of realistic decompositions in the distribution. Note that in practice the approximate distributions $q(\mathbf{z})$ is a parametric form of ω (variational parameters) which characterize this distribution; so it is more accurate to denote it by $q_{\mathbf{Z}}(\mathbf{z};\omega)$, in which ω denote set of variational parameters.

R Sometimes this method is called *meanfield* variational method; this name is because of decomposition of $q(.)$. Note that even without decomposition assumptions here, one could follow all of the following derivations. When using the final results in updating, decomposition will help us to find easier update rules.

To approximate the true posterior $p(.)$ using the decomposed distribution $q(.)$ we should find a measure of distance between two functions, and also is practical in computational sense[1].

■ **Lemma 13.1** We have the following fact,

$$\log p(\mathbf{X}|\theta) = \mathscr{L}(q_{\mathbf{Z}}) + \mathrm{KL}(q_{\mathbf{Z}}||p_{\mathbf{Z}|\mathbf{X}}). \tag{13.3}$$

In which we defined the following two notations; the first one is KL-*divergence*,

$$\mathrm{KL}(q_{\mathbf{Z}}||p_{\mathbf{Z}|\mathbf{X}}) \triangleq \mathbb{E}_q\left[\log \frac{q(\mathbf{Z})}{p(\mathbf{Z}|\mathbf{X},\theta)}\right] \tag{13.4}$$

[1] In practical sense, this is more important! This is the reason for long domination of the bloodthirsty quadratic error!!

And the second one is the lower bound on the likelihood (to be shown)

$$\mathscr{L}(q,\theta) \triangleq \int_{\mathbb{R}^m} q(\mathbf{Z}) \log \frac{p(\mathbf{X},\mathbf{Z}|\theta)}{q(\mathbf{Z})} d\mathbf{Z} \tag{13.5}$$

Proof. Now using equation 13.1 we have,

$$\log p(\mathbf{X}|\theta) = \log p(\mathbf{X},\mathbf{Z}|\theta) - \log p(\mathbf{Z}|\mathbf{X},\theta)$$
$$\log p(\mathbf{X}|\theta) = \log \frac{p(\mathbf{X},\mathbf{Z}|\theta)}{q(\mathbf{Z})} - \log \frac{p(\mathbf{Z}|\mathbf{X},\theta)}{q(\mathbf{Z})}.$$

Multiplying two sides in $q(\mathbf{Z})$ we have,

$$q(\mathbf{Z}) \log p(\mathbf{X}|\theta) = q(\mathbf{Z}) \log \frac{p(\mathbf{X},\mathbf{Z}|\theta)}{q(\mathbf{Z})} - q(\mathbf{z}) \log \frac{p(\mathbf{Z}|\mathbf{X},\theta)}{q(\mathbf{Z})}$$
$$\int_{\mathbb{R}^m} q(\mathbf{Z}) \log p(\mathbf{X}|\theta) d\mathbf{Z} = \int_{\mathbb{R}^m} q(\mathbf{Z}) \log \frac{p(\mathbf{X},\mathbf{Z}|\theta)}{q(\mathbf{Z})} d\mathbf{Z} - \int_{\mathbb{R}^m} q(\mathbf{Z}) \log \frac{p(\mathbf{Z}|\mathbf{X},\theta)}{q(\mathbf{Z})} d\mathbf{Z}.$$

Note that in the left part of the above equation $p(\mathbf{X}|\theta)$ is not a function of $\mathbf{z}$ and thus, $\int_{\mathbb{R}^m} q(\mathbf{Z}) \log p_{\mathbf{X}}(\mathbf{X}|\theta) d\mathbf{Z} = \log p(\mathbf{X}|\theta)$. Note that,

$$\mathrm{KL}(q_{\mathbf{Z}}||p_{\mathbf{Z}|\mathbf{X}}) \triangleq -\int_{\mathbb{R}^m} q(\mathbf{Z}) \log \frac{p(\mathbf{Z}|\mathbf{X},\theta)}{q(\mathbf{Z})} d\mathbf{Z} \tag{13.6}$$
$$= \int_{\mathbb{R}^m} q(\mathbf{Z}) \log \frac{q(\mathbf{Z})}{p(\mathbf{Z}|\mathbf{X},\theta)} d\mathbf{z} \tag{13.7}$$
$$= \mathbb{E}_q \left[\log \frac{q(\mathbf{Z})}{p(\mathbf{Z}|\mathbf{X},\theta)} \right] \tag{13.8}$$

Putting the results together we have,

$$\log p(\mathbf{X}|\theta) = \mathscr{L}(q_{\mathbf{Z}}) + \mathrm{KL}(q_{\mathbf{Z}}||p_{\mathbf{Z}|\mathbf{X}}). \tag{13.9}$$

■

The equation (16.1) is shown in Figure (13.2). Now in equation 16.1 note that left side of the

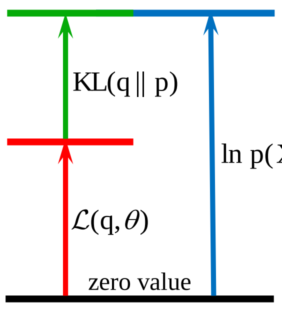


Figure 13.2: Representation of equation 16.1.

equation is not a function of $q(.)$. The training consists of increasing the overall likelihood we are doing a two step procedure similar to EM algorithm:

1. Initialize θ, and variational parameters of $q(.)$.
2. Repeat until convergence

(a) E-step:

$$q^{(t+1)} = \arg\max_{q} \mathscr{L}(q^{(t)}, \theta^{(t)}) \tag{13.10}$$

(b) M-step:

$$\theta^{(t+1)} = \arg\max_{\theta} \mathscr{L}(q^{(t+1)}, \theta^{(t)}) = \arg\min_{\theta} \mathrm{KL}(q_{\mathbf{Z}} || p_{\mathbf{Z}|\mathbf{X}}) \tag{13.11}$$

Visualization of these two steps is shown in Figure 13.3.

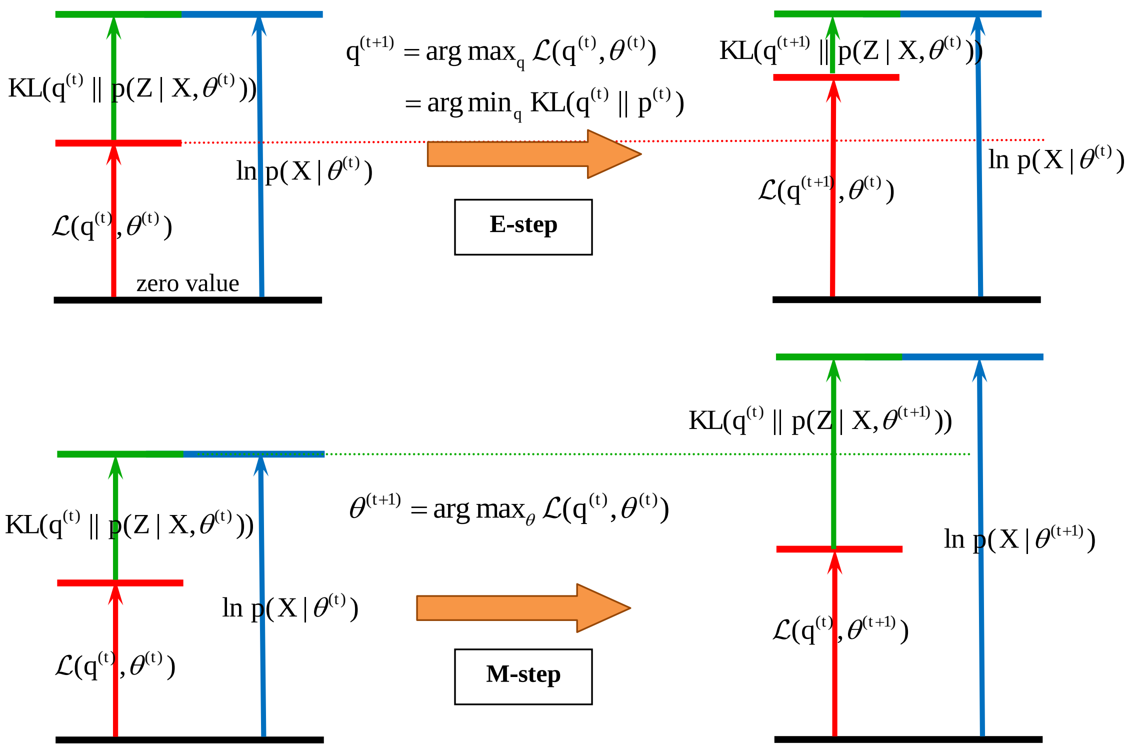

KL(q^(t) || p(Z | X, θ^(t)))
ℒ(q^(t), θ^(t))
zero value
ln p(X | θ^(t))
q^(t+1) = arg max_q ℒ(q^(t), θ^(t))
= arg min_q KL(q^(t) || p^(t))
E-step
KL(q^(t+1) || p(Z | X, θ^(t)))
ℒ(q^(t+1), θ^(t))
ln p(X | θ^(t))
KL(q^(t) || p(Z | X, θ^(t)))
ℒ(q^(t), θ^(t))
ln p(X | θ^(t))
θ^(t+1) = arg max_θ ℒ(q^(t), θ^(t))
M-step
KL(q^(t) || p(Z | X, θ^(t+1)))
ℒ(q^(t), θ^(t+1))
ln p(X | θ^(t+1))

Figure 13.3: Steps in variational learning; the above part is E-step and the other is M-step.

R Basically we are doing *coordinate ascent* optimization, i.e. given one multivariable objective function, fix all of the variables but one, and maximize with respect to that variable.

13.2 Yet another justification

Lemma 13.2 The $\mathscr{L}(q, \theta)$ is a lower bound on the likelihood distribution. In other words,

$$p(\mathbf{X}|\theta) \geq \exp[\mathscr{L}(q, \theta)]$$

Proof. I use Jensen's inequality here to show that $\mathscr{L}(q, \theta)$ a lower bound for the original likelihood:

$$
\begin{aligned}
\ln p(\mathbf{X}|\theta) &= \ln \int_{\mathbb{R}^m} p(\mathbf{X},\mathbf{Z}|\theta) d\mathbf{Z} \\
&= \ln \int_{\mathbb{R}^m} q(\mathbf{Z}) \frac{p(\mathbf{X},\mathbf{Z}|\theta)}{q(\mathbf{Z})} d\mathbf{Z} \\
&\geq \int_{\mathbb{R}^m} q(\mathbf{Z}) \ln \frac{p(\mathbf{X},\mathbf{Z}|\theta)}{q(\mathbf{Z})} d\mathbf{Z} = \mathscr{L}(q,\theta)
\end{aligned}
$$

This shows that $\exp[\mathscr{L}(q,\theta)]$ a lower bound for likelihood $p(\mathbf{X}|\theta)$:

$$\Rightarrow p(\mathbf{X}|\theta) \geq \exp[\mathscr{L}(q,\theta)]$$

■

Corollary 13.1 Maximizing $\mathscr{L}(q,\theta)$ (lower bound) will result in maximizing the likelihood $p(\mathbf{X}|\theta)$. For this reason, $\mathscr{L}(q,\theta)$ is sometimes called **ELBO** (Evidence Lower Bound).

Another useful point in the above result is that, one could use the lower bound on likelihood as a good measure of convergence, instead of main likelihood, in case it has a complicated form.

R However through lots of examples people have shown effectiveness of maximizing lower bound on the likelihood (instead of maximizing likelihood itself), indeed it is a serious question that when this argument is true and when is not. This topic of some ongoing research in community.

Proposition 13.1 — Convexity of the lower bound. The ELBO bound is convex with respect to each of the $q(Z_i)$ (proof?).

13.3 More simplification of updates for mean-field family

Let's consider the equation 16.2,

$$
\begin{aligned}
\mathscr{L}(q,\theta) &= \int_{\mathbb{R}^m} q(\mathbf{Z}) \log \frac{p(\mathbf{X},\mathbf{Z}|\theta)}{q(\mathbf{Z})} d\mathbf{Z} \\
&= \int_{\mathbb{R}^m} q(\mathbf{Z}) \log p(\mathbf{X},\mathbf{Z}|\theta) d\mathbf{Z} - \int_{\mathbb{R}^m} q(\mathbf{Z}) \log q(\mathbf{Z}) d\mathbf{Z}
\end{aligned}
$$

R The second term in the above final relation is *information entropy* of $q(\mathbf{Z})$.

$$
\begin{aligned}
\mathscr{L}(q,\theta) &= \mathbb{E}_q[\log p(\mathbf{X},\mathbf{Z}|\theta) - \log q(\mathbf{Z})] \\
&= \mathbb{E}_q\left[\log p(\mathbf{X},\mathbf{Z}|\theta) - \log \prod_{k=1}^{m} q(Z_k)\right] \\
&= \mathbb{E}_q[\log p(\mathbf{X},\mathbf{Z}|\theta)] - \sum_{k=1}^{m} \mathbb{E}_q[\log q(Z_k)]
\end{aligned}
$$

Now lets say we want to optimize the above expression with respect to one term in approximate posterior, say $q(Z_j)$; thus, we try to take the corresponding term out of the whole equation,

$$\begin{aligned}
\mathscr{L}(q,\theta) &= \mathbb{E}_{q(\mathbf{Z})}\left[\log p(\mathbf{X},\mathbf{Z}|\theta)\right] - \sum_{k=1}^{m}\mathbb{E}_{q(Z_k)}\left[\log q(Z_k)\right] \\
&= \mathbb{E}_{q(Z_j)}\left[\mathbb{E}_{\prod_{\substack{i=1\\i\neq j}}^{m} q(Z_i)}\left[\log p(\mathbf{X},\mathbf{Z}|\theta)\right]\right] - \sum_{k=1}^{m}\mathbb{E}_{q(Z_k)}\left[\log q(Z_k)\right] \\
&= \mathbb{E}_{q(Z_j)}\left[\mathbb{E}_{\prod_{\substack{i=1\\i\neq j}}^{m} q(Z_i)}\left[\log p(\mathbf{X},\mathbf{Z}|\theta)\right]\right] - \mathbb{E}_{q(Z_j)}\left[\log q(Z_j)\right] - \sum_{\substack{k=1\\k\neq j}}^{m}\mathbb{E}_{q(Z_k)}\left[\log q(Z_k)\right] \\
&= \mathbb{E}_{q(Z_j)}\left[\log\left(\exp\left\{\mathbb{E}_{\prod_{\substack{i=1\\i\neq j}}^{m} q(Z_i)}\left[\log p(\mathbf{X},\mathbf{Z}|\theta)\right]\right\}\right)\right] - \mathbb{E}_{q(Z_j)}\left[\log q(Z_j)\right] - \sum_{\substack{k=1\\k\neq j}}^{m}\mathbb{E}_{q(Z_k)}\left[\log q(Z_k)\right] \\
&= \mathbb{E}_{q(Z_j)}\left[\log\left(\exp\left\{\mathbb{E}_{\prod_{\substack{i=1\\i\neq j}}^{m} q(Z_i)}\left[\log p(\mathbf{X},\mathbf{Z}|\theta)\right]\right\}\right) - \log q(Z_j)\right] - \sum_{\substack{k=1\\k\neq j}}^{m}\mathbb{E}_{q(Z_k)}\left[\log q(Z_k)\right] \\
&= \mathbb{E}_{q(Z_j)}\left[\log\frac{\exp\left\{\mathbb{E}_{\prod_{\substack{i=1\\i\neq j}}^{m} q_{Z_i}}\left[\log p(\mathbf{X},\mathbf{Z}|\theta)\right]\right\}}{q(Z_j)}\right] - \sum_{\substack{k=1\\k\neq j}}^{m}\mathbb{E}_{q(Z_k)}\left[\log q(Z_k)\right] \\
&= -\mathrm{KL}\left(q(Z_j)||\exp\left\{\mathbb{E}_{\prod_{\substack{i=1\\i\neq j}}^{m} q(Z_i)}\left[\log p(\mathbf{X},\mathbf{Z}|\theta)\right]\right\}\right) - \sum_{\substack{k=1\\k\neq j}}^{m}\mathbb{E}_{q(Z_k)}\left[\log q(Z_k)\right]
\end{aligned}$$

Now only the left term is a function of $q(Z_j)$; to maximize $\mathscr{L}(q,\theta)$ with respect to $q(Z_j)$, we should minimize the KL-*divergence*, which results in the following,

$$q(Z_j) \propto \exp\left\{\mathbb{E}_{q(Z_{-j})}\left[\log p(\mathbf{X},\mathbf{Z}|\theta)\right]\right\} \tag{13.12}$$

where $q(Z_{-j})$ stands for $\prod_{\substack{i=1\\i\neq j}}^{m} q(Z_i)$.

R There is a very good similarities between Gibbs sampling and mean-field variational learning. In Gibbs sampling we sample from conditionals (function of only one variable); this is similar to coordinate ascent updates in variational learning.

13.4 On minimization of divergence measures

Back to the variational M-step in Equation 13.11, it is just a mapping, from distribution $p(.)$ to a (tractable) family of distributions $q(.)$. In addition to the KL-*divergence*, this mapping could be done via many other divergence measures between functions, to see examples and some properties see the Wiki pages for KL-*divergence*, Bregman divergence or f-divergence.

There is a nice discussion of divergence minimization in Altun and Smola (2006). Here we briefly mention some of the important results.

13.5 Variational learning with exponential family

Let us first review some properties about the exponential families. If you are already familiar with the exponential family, skip the definition.

Definition 13.1 — exponential families. We want to define a probability distribution on $\mathscr{X}$ domain. Assume the vector of *sufficient estimators* $\phi(\mathbf{x}) = [\phi(\mathbf{x})_1, \ldots, \phi(\mathbf{x})_d]^\top$ in in which $\phi_i(\mathbf{x}) : \mathscr{X} \to \mathbb{R}$ is a function defined for any $j = 1, \ldots, d$. This can correspond to the all nodes and connections in a Graphical model. In addition assume the vector of *canonical parameters* $\theta = [\theta_1, \ldots, \theta_d]^\top$, in which $\theta_i \in \mathbb{R}$. Now define the distribution as following,

$$p_\theta(\mathbf{x}) = h(\mathbf{x}) \exp\left(\theta^\top \phi(\mathbf{x}) - A(\theta)\right)$$

where $h(\mathbf{x})$ is an arbitrary function of $\mathbf{x}$. Usually this is $h(\mathbf{x}) = 1$. $A(\theta)$ is the *cumulative generating function*, and is defines as $A(\theta) = \log \int_{\mathbf{x}\in\mathscr{X}} h(\mathbf{x}) \exp\left(\theta^\top \phi(\mathbf{x})\right) \nu(d\mathbf{x})$, where $\nu(.)$ is the measure defined on $\mathscr{X}$. Note that the normalizing function for the distribution is defined as $\mathscr{Z}(\theta) = \int_{\mathbf{x}\in\mathscr{X}} h(\mathbf{x}) \exp\left(\theta^\top \phi(\mathbf{x})\right) \nu(d\mathbf{x}) = \log A(\theta)$.

The distribution is called *minimal* if the set of sufficient estimators are linearly independent. In other words, there is no $\theta \in \mathbb{R}^n$ for which $\theta^\top \phi(\mathbf{x}) = 0$ for all $x \in \mathscr{X}$. If the distribution is not regular, it is called *over-complete*. The distribution is called *regular* if for any vector coefficients the distribution is normalizable(a valid distribution). In other words, if we define the set $\Omega = \{\theta \in \mathbb{R}^n | \ |A(\theta)| < +\infty\}$.

This family of distributions has different properties, some of which we are listing here. Many important distributions lie in the exponential family, for example Gaussian, Multinomial and Bernoulli. There is a rich table for transformation of many distributions can be found on the Wiki. In addition to the parametric distributions mentioned in the Wiki, it can be shown that any joint probability distribution on discrete random variables can be transformed into the exponential form.

The *cumulative generating function* has very important properties. The first derivative of commutative generating function is the expectation of the vector of the sufficient statistics,

$$\nabla_\theta A(\theta) = \mathbb{E}_p(\phi(\mathbf{x})). \tag{13.13}$$

This property is very useful since we can compute the integral of expectation, by differentiation. There is a similar relation for the second derivative,

$$\nabla^2_\theta A(\theta) = \mathrm{Var}_p(\phi(\mathbf{x})).$$

The *sufficiency* is a statistical property, and it means that “having the values of the sufficient statistics, we don’t need the data points to do inference". In other words, after learning model, we can just through away the training data points. This is helpful because usually the dimension of the data points are much higher than the number of the sufficient statistics. Also if you consider the joint distribution of IID variables (likelihood), $p_\theta(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \prod_{i=1}^n h(\mathbf{x}_i) \exp\left(\theta^\top \sum_{i=1}^n \phi(\mathbf{x}_i) - nA(\theta)\right)$. This shows the only information visible from the training random variables is the summation $\sum_{i=1}^n \phi(\mathbf{x}_i)$. Thus, have the value of this summation is enough for inference using this exponential model. The exact definition of sufficiency can be explained using the FisherNeyman factorization theorem (see Wiki).

Continuing the previous note, let's find the maximum likelihood estimation of the parameter vector θ using the likelihood of the IID random variable observations,

$$\theta_{ML} = \arg\max_{\theta} p_{\theta}(\mathbf{x}_1, \ldots, \mathbf{x}_n).$$

Putting the derivative of the joint distribution with respect to parameters to zero, we find the maximum-likeli estimation, $\nabla_{\theta} A(\theta) = \frac{1}{n}\sum_{i=1}^{n} \phi(\mathbf{x}_i)$. Using our previous findings, we see that $\mathbb{E}_p(\phi(\mathbf{x})) = \frac{1}{n}\sum_{i=1}^{n} \phi(\mathbf{x}_i)$ which says that the expected value of the sufficient statistic matches its average found using IID samples (by having enough data). This is very useful since in finding the expected value of the sufficient statistics by averaging their values from the training data, there is no need for parameter estimation for the original distribution.

Another very important property is the *convexity* of $A(\theta)$ with respect to θ. In other words,

$$A(\beta\theta_1 + (1-\beta)\theta_2) < \beta A(\theta_1) + (1-\beta)A(\theta_2), \forall \beta \in (0,1), \theta_1 \neq \theta_2$$

This is easy to prove; just observe that the second derivative (Hessian) of $A(\theta)$ equals to is the covariance function of the sufficient estimators, and is a positive-semi-definite matrix(if you don't know why see Wiki).

Another nice property of the exponential family is the *conjugacy*. This means that, if we choose an exponential prior distribution in a Bayesian model with an exponential likelihood, the posterior distribution will be in the same exponential family.

Proof. This is easy to show; say the output observation is $\mathbf{x}$ with parameter vector of θ with exponential distribution,

$$p(\mathbf{x}|\eta) = h(\mathbf{x}) \exp\left(\theta(\eta)^{\top}\phi(\mathbf{x}) - A(\theta(\eta))\right),$$

which is the definition we used for the exponential family with the difference that the vector of parameters is function of another random variable η. The prior over the η we define to be,

$$p_{\lambda}(\eta) = g(\eta) \exp\left(\theta(\eta)^{\top}\lambda - B(\lambda)\right).$$

Let's assume we have a matrix of observation $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$, and the likelihood for the random IID observation variables is,

$$p(\mathbf{X}|\eta) = \prod_{i=1}^{n} h(\mathbf{x}_i) \exp\left(\theta(\eta)^{\top}\sum_{i=1}^{n}\phi(\mathbf{x}_i) - nA(\theta(\eta))\right).$$

Using the Bayes formula we know that posterior equals to $p(\eta|\mathbf{X};\lambda) \propto p(\mathbf{X}|\eta)p(\eta;\lambda)$(Note that here I am assuming that η is a vector of random variables(not constant) but λ is a vector of parameters; that's why I write $p(\eta;\lambda)$ instead of $p(\eta|\lambda)$). Using the Bayes formula, the posterior is following,

$$p(\eta|\mathbf{X};\lambda) \propto \exp\left(\theta(\eta)^{\top}\left(\lambda + \sum_{i=1}^{n}\phi(\mathbf{x}_i)\right) - (nA(\theta(\eta)) + B(\lambda))\right).$$

■

The exponential distribution also arises naturally from the *maximum entropy principle*. The maximum entropy procedure consists of seeking the probability distribution which maximizes information entropy, subject to the constraints of the information. If we constrain the expected values of the sufficient statistics to be mean of the empirical mean of them under the sampled data, the resulting distribution which maximizes the entropy is the exponential family. To make the statement more accurate, let's express it in terms of formula. Given iid random variables, $\mathbf{x}_1, \ldots, \mathbf{x}_n \sim p^*$, where p^* is the underlying *unknown* distribution. Define a set of functions $\{\phi_i(.) : \mathscr{X} \to \mathbb{R}\}_{i=1}^{d}$, and consider the empirical mean of the sampled data under these functions,

$$\hat{\mu}_j = \frac{1}{n} \sum_{i=1}^{n} \phi_j(\mathbf{x}_i), \ \forall j \in \{1, \ldots, d\}$$

The problem is that we are looking for a distribution, $p(.)$ which has the same set of expected values for the defined functions as their empirical distribution, i.e.

$$\mathbb{E}_p [\phi_j(\mathbf{x})] = \int_{\mathscr{X}} \phi_j(\mathbf{x}) p(\mathbf{x}) \nu(d\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \phi_j(\mathbf{x}_i) = \hat{\mu}_j,$$

and at the same time maximizes the entropy under this distribution,

$$E = \int_{\mathscr{X}} -p(\mathbf{x}) \log p(\mathbf{x}) \nu(d\mathbf{x}).$$

This constrained optimization will result in the parametric form of the exponential family of distributions defined here.

Looking at the update rule at equation 13.12, one might suspect that we probably can devise easier updates for each $q(Z_i)$, when they are from exponential families. In fact, having only a few moments of exponents in exponential families are *sufficient* estimators of this family of distributions. Let's say 1 has the following general form in exponential families,

$$p(\mathbf{X}, \mathbf{Z}|\theta) = f(\chi, \nu) \exp\left\{\eta(\mathbf{X}, \mathbf{Z})^{\mathrm{T}} \chi - \nu A(\eta(\mathbf{X}, \mathbf{Z}))\right\}$$

$$\ln p(\mathbf{X}, \mathbf{Z}|\theta) = \ln f(\chi, \nu) + \eta(\mathbf{X}, \mathbf{Z})^{\mathrm{T}} \chi - \nu A(\eta(\mathbf{X}, \mathbf{Z}))$$

$$\mathbb{E}_{q(Z_{-j})} \ln p(\mathbf{X}, \mathbf{Z}|\theta) = \ln f(\chi, \nu) + \mathbb{E}_{q(Z_{-j})} \eta(\mathbf{X}, \mathbf{Z})^{\mathrm{T}} \chi - \mathbb{E}_{q(Z_{-j})} \nu A(\eta(\mathbf{X}, \mathbf{Z}))$$

then,

$$q(Z_j) \propto f(\chi, \nu) \exp\left\{\mathbb{E}_{q(Z_{-j})} \eta(\mathbf{X}, \mathbf{Z})^{\mathrm{T}} \chi\right\}$$

This shows that that choosing the approximating functions $q(\mathbf{Z})$, from the same exponential family of the conditional $p(\mathbf{X}, \mathbf{Z}|\theta)$ would crucially help in E-step updates; we only need to calculate expectation of the exponent with respect to approximating family. More on this property and more examples at Wainwright and Jordan (2008).

13.5.1 Mean parametrization and marginal polytopes

Let's assume the probability density $p(.)$ (not necessarily in the exponential family). Assume a set of local functions $\{\phi_i(.) : \mathscr{X} \to \mathbb{R}\}_{i \in \mathscr{I}}$, in which I is a set of indices. Also a vector of mean parameters $\mu = [\mu_1, \ldots, \mu_d]^\top$ which are defined as following,

$$\mu_j = \mathbb{E}_p [\phi_j(\mathbf{x})] = \int_{\mathscr{X}} \phi_j(\mathbf{x}) p(\mathbf{x}) \nu(d\mathbf{x})$$

Now we define the whole space of realizable mean vectors by any families of distributions $\mathscr{P}$ (not limited to exponential family) as *marginal polytope*,

$$\mathscr{M} = \left\{\mu \in \mathbb{R}^{|\mathscr{I}|} | \exists p(.) \in \mathscr{P} \text{ s.t. } \mathbb{E}_p[\phi(\mathbf{x})] = \mu\right\}$$

If in the above definition we limit the family of the distributions $\mathscr{P}$ to the exponential family $\mathscr{E}$, since the exponential family is a strict the general distributions, the resulting polytope $\mathscr{M}^0$ is a strict subset of the polytope $\mathscr{M}$,

$$\mathscr{M}^0 = \left\{\mu \in \mathbb{R}^{|\mathscr{I}|} | \exists p(.) \in \mathscr{E} \text{ s.t. } \mathbb{E}_p[\phi(\mathbf{x})] = \mu\right\} \subset \mathscr{M}. \tag{13.14}$$

Proposition 13.2 The marginal polytope $\mathscr{M}$ is a convex subset of $\mathbb{R}^{|\mathscr{I}|}$ (proof?).

■ **Example 13.2 — A simple Gaussian MRF.** Let's assume $\phi(x) = [x,\ x^2]^\top$, and the vector of means $[\mu_1,\ \mu_2]$ realized under any arbitrary distribution $p(.)$ such that $[\mu_1,\ \mu_2] = \left[\mathbb{E}_p[x],\ \mathbb{E}_p\left[x^2\right]\right]$. The only constrain in such realization is that $\mathbb{V}(x) = \mathbb{E}_p\left[x^2\right] - \mathbb{E}_p^2[x] = \mu_2 - \mu_1^2 \geq 0$ ■

■ **Example 13.3 — General Gaussian MRF.** Assume a graph, on which each vertex is associated with a continuous variable on $\mathbb{R}$ and the variables which are connected by vertices are dependent based on a Gaussian distribution. Let's assume connection between all of the variables (a complete graph, k_n). It is more convenient to denote the sufficient statistics in a matrix instead of a vector, since their pairwise interaction (a quadratic model) is important,

$$\phi(\mathbf{x}) = \begin{bmatrix} 1 & x_1 & x_2 & \dots & x_n \\ x_1 & x_1^2 & x_1x_2 & \dots & x_1x_n \\ x_2 & x_2x_1 & x_2^2 & \dots & x_2x_n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_n & x_1x_n & x_2x_n & \dots & x_n^2 \end{bmatrix}.$$

Given these local functions, we want to find the possible mean values such that $\mathbb{E}_p[\phi(\mathbf{x})] = \mu$. We show the corresponding mean matrix as follows,

$$U(\mu) = \begin{bmatrix} 1 & \mu_1 & \mu_2 & \dots & \mu_n \\ \mu_1 & \mu_{11} & \mu_{12} & \dots & \mu_{1n} \\ \mu_2 & \mu_{21} & \mu_{22} & \dots & \mu_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mu_n & \mu_{1n} & \mu_{2n} & \dots & \mu_{nn} \end{bmatrix}.$$

Now all of the constraints on the mean-space for this model could be found by setting $U(\mu)$ be positive definite,

$$\mathscr{M}_{\text{Gaussian-MRF}} = \left\{\mu \in \mathbb{R}^{n+\binom{n}{2}} | U(\mu) \succeq 0\right\}$$

■

■ **Example 13.4 — A simple Ising model.** Let's assume a very simple Ising models with only two random variables on a graph with two vertices which are connected by an edge. Then the local functions are $\phi(\mathbf{x}) = [x_1, x_2, x_1x_2]^\top$. Now we want to find the space of all $[\mu_1, \mu_2, \mu_3]$ such that, $[\mu_1, \mu_2, \mu_3] = [\mathbb{E}[x_1], \mathbb{E}[x_2], \mathbb{E}[x_1x_2]]$. If we simplify this, $\mathbb{E}[x_1] = 1 \times \mathbb{P}(x_1 = 1) + 0 \times \mathbb{P}(x_1 = 0) = \mathbb{P}(x_1 = 1)$, similarly $\mathbb{E}[x_2] = \mathbb{P}(x_2 = 1)$ and $\mathbb{E}[x_1x_2] =$

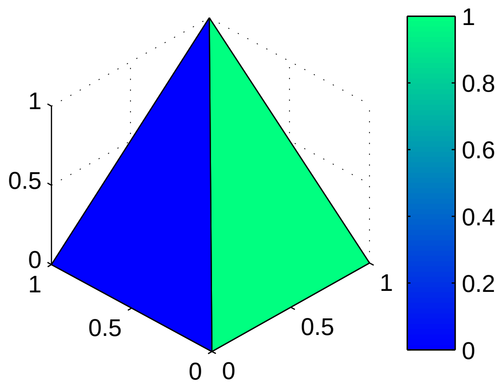


Figure 13.4: The marginal polytope for the Ising model in the example.

$1 \times 1 \times \mathbb{P}(x_1 = 1, x_2 = 1) + 0 \times 1 \times \ldots + \ldots = \mathbb{P}(x_1 = 1, x_2 = 1)$. Each of the individual probabilities can vary between zero and 1. Also the joint distribution $\mathbb{P}(x_1 = 1, x_2 = 1)$ is strictly less than each of the other individual distributions. Thus the resulting polytope is depicted in Figure 13.4. ■

13.5.2 Convex dualities

Here want to provide another view of variational inference. Let us assume an arbitrary function $f(u)$ which is defined on $u \in \mathbb{R}^n$. We show the *conjugate dual* of $f(.)$ by $f^*(.)$ and define this function as following:

$$f^*(v) = \sup_{u \in \mathbb{R}^n} \{\langle u, v\rangle - f(u)\}$$

for any $v \in \mathbb{R}^n$. In general the above definition holds in any domain that the Lebesgue measure holds (or on other words the inner product $\langle u, v\rangle$ makes sense) and is not limited to real numbers.

R Re-using the same definition above we find the following,

$$(f^*)^*(u) = \sup_{v \in \mathbb{R}^n} \{\langle u, v\rangle - f^*(v)\}$$

gives the double conjugate dual of the function. In certain conditions, the double conjugate dual of a function equals to itself. If the function $f(.)$ is *well-behaved (i.e. convex and semi-continuous)* then the double conjugate dual of a function equals to itself:

$$\begin{cases} f^*(v) = \sup_{u \in \mathbb{R}^n} \{\langle u, v\rangle - f(u)\} \\ f(u) = \sup_{v \in \mathbb{R}^n} \{\langle u, v\rangle - f^*(v)\} \end{cases}$$

which shows the strong coupling between the functions and its conjugate dual. If the function is *concave* the same duality holds with inf(.) operator, instead of sup(.).

$$\begin{cases} f^*(v) = \inf_{u \in \mathbb{R}^n} \{\langle u, v\rangle - f(u)\} \\ f(u) = \inf_{v \in \mathbb{R}^n} \{\langle u, v\rangle - f^*(v)\} \end{cases}$$

This is called **Fenchel's duality theorem**.

The reason for introducing these dualities is that, sometimes the optimization procedure is harder than optimization in its conjugate dual form (or vice versa). Thus we can use these conjugate dualities to find easier optimization schemes, as it will be shown.

Proposition 13.3 The conjugate dual function is always a convex function (proof?).

13.5.3 The log-partition function and conjugate duality

Let's instead of the general functions $f(.)$, assume the log-partition function (cummulant function).

$$A^*(\mu) = \sup_{\mu \in \Omega} \left\{ \theta^\top \mu - A(\mu) \right\}.$$

As in the Equation 13.13 we have shown that $\nabla_\theta A(\theta) = \mathbb{E}_p(\phi(\mathbf{x}))$. We can consider $\nabla_\theta A(\theta) : \Omega \to \mathscr{M}$ as a function which maps from the parameter space to the mean space, and we call this a *forward mapping*. Thus, because $A(\theta)$ is defined for the exponential family, this forward mapping, maps all possible parameters to all possible means by these parameters in the exponential family which we called $\mathscr{M}^0$ in Equation 13.14. Thus the mapping using $\nabla_\theta A(\theta) : \Omega$ covers the whole $\mathscr{M}^0$ which is a strict subset of $\mathscr{M}$. Thus, there might be some elements in $\mu \in \mathscr{M} \setminus \mathscr{M}^0$ which are not realizable by any parameter in the exponential family. If we limit the domain to $\mathscr{M}^0$ it is easy to show that the mapping is one-to-one if the exponentials are minimal (easy to show by contradiction).

Proposition 13.4 The mapping $\nabla_\theta A(\theta) : \Omega \to \mathscr{M}^0$ is one-to-one if and only if the exponential distribution is minimal.

The *backward mapping* is defined in the similar way to the forward mapping; For minimal exponential family, for any $\mu \in \mathscr{M}^0$, there exists a $\theta \in \Omega$ such that $\mathbb{E}_{p_\theta}(\phi(\mathbf{x})) = \mu$. Note that among non-exponential families there might be some other other distributions that have the same mean μ, but all of them have less entropy that the exponential one, since the exponential family has the maximum entropy, given the means as constrains.

For any $\mu \in \mathscr{M}^0$, let θ be the corresponding parameter based on the equation $\mathbb{E}_{p_\theta}(\phi(\mathbf{x})) = \mu$. We can show that the dual function takes the following form,

$$A^*(\mu) = \begin{cases} -H(p_\theta),\ \mu \in \mathscr{M}^0 \\ +\infty \qquad \text{otherwise} \end{cases}$$

in which $H(p_\theta)$ is the entropy of the distribution p_θ. This property is very useful when using the maximum-entropy rule for model-selection, since it gives the direct connection to the entropy of the model. At the same time, given the parameters of the model, calculating the double-conjugate (the cumlant itself) will give the corresponding vector of means which is as if *inference* given a model. This explanation should prove the importance of the variational conjugate dualities and their usefulness in inference and model-selection problems.

■ **Example 13.5 — Conjugate duality on a Bernoulli distribution.** The Bernoulli distribution PMF is defined as $p(x) = \beta^x(1-\beta)^{1-x}$, over random variable $x \in \{0,1\}$ where β is the probability of success in one toss. By a little modification we can change the representation to exponential form:

$$\begin{aligned} p(x) &= \beta^x(1-\beta)^{1-x} \\ &= \exp\left\{\log\left[\beta^x(1-\beta)^{1-x}\right]\right\} \\ &= \exp\left(x\log\beta + (1-x)\log(1-\beta)\right) \\ &\propto \exp(x\theta),\ A(\theta) = \log(1+\exp(\theta)) \end{aligned}$$

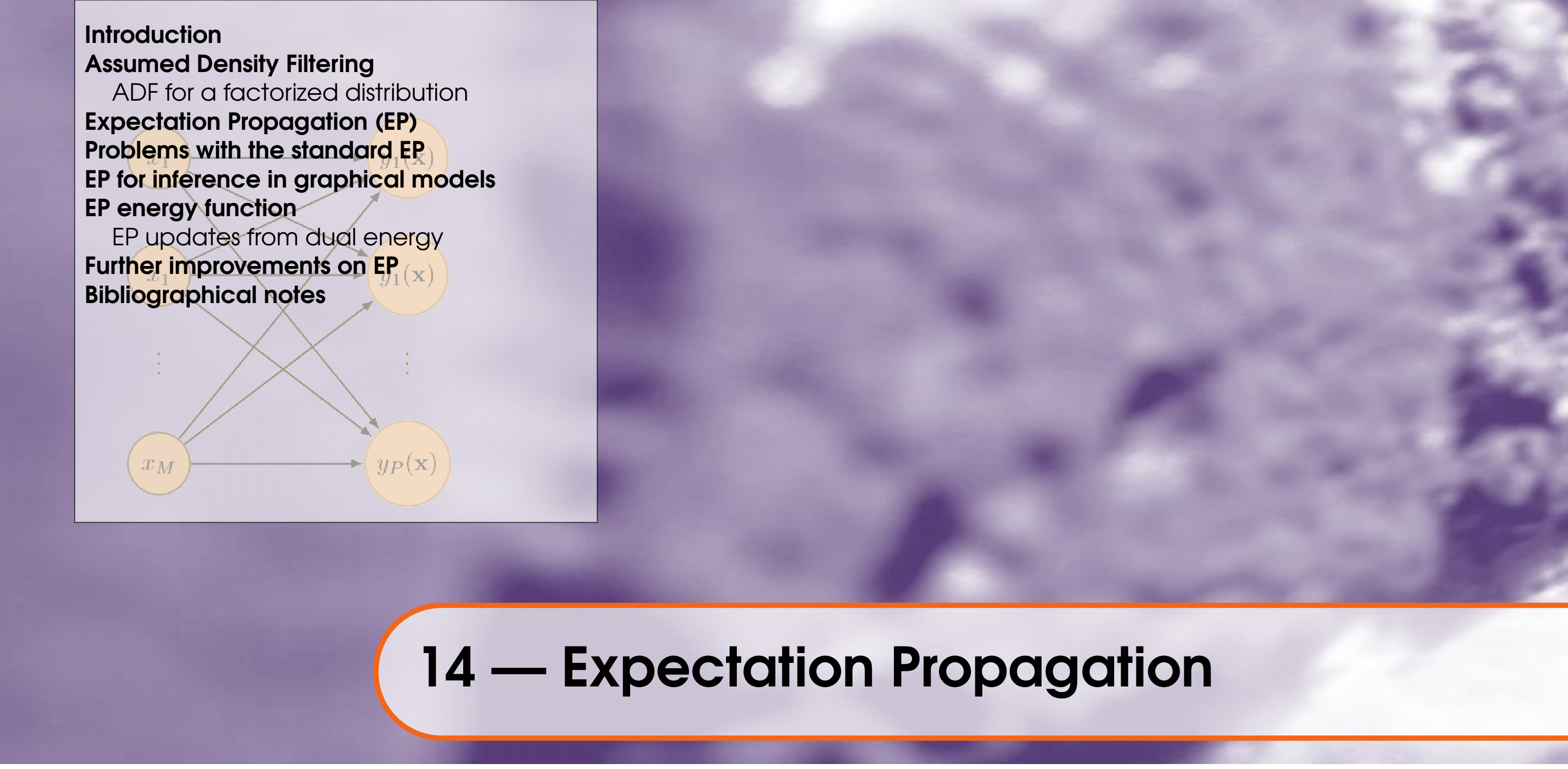
where $\theta = \log \frac{\beta}{1+\beta}$. We define $A^*(\mu) = \sup_{\mu \in \mathbb{R}} \{\theta\mu - \log(1+\exp(\theta))\}$.. Simplifying this equation for any $\mu \in (0,1)$ we find the that $A^*(\mu) = \mu \log \mu + (1-\mu)\log(1-\mu)$ which is the entropy for the Bernoulli distribution as mentioned before.

■

13.6 Bibliographical notes

In preparation of this part I have used Bishop (2006). At some visualizations I've also been inspired by Julia Hockenmaier's slides [2]. Some examples and ideas are also inspired from the lecture notesat Cevher (2008). An important relevant paper is Wainwright and Jordan (2008) which has given the inspiration in many of the notations and examples.

[2] http://courses.engr.illinois.edu/cs598jhm/



14 — Expectation Propagation

14.1 Introduction

Expectation Propagation(EP) is one of approaches for approximate inference which first formulated the way we see today at Minka (2001b) though the idea has roots in many previous works in various areas. It can be considered as a variant of message-passing where each of the individual messages are approximated while being transferred. To introduce EP, it is easier to first start with a couple of approximations (projections) for any arbitrary distribution. Here we start with Assumed Density Filtering(ADF).

14.2 Assumed Density Filtering

ADF is introduced independently in several areas at different times under different names like "Moment Matching", "Weak Marginalization", etc Lauritzen (1992); Koller et al. (1999); Maybeck (1979). The idea used is so much similar to the update equations in Kalman Filtering.
Assume that using the $\mathbf{x}$ as observations we want to make inference about the latent variables $\mathbf{y}$. Now the goal is to find a an exact posterior $p(\mathbf{y}|\mathbf{x})$ and only keep the approximation to it $q(\mathbf{y})$, using a tractable form, say exponential form, and possibly use it for future calculation. This further approximation of the posterior is can be seen as *projection* of one distribution, over another family of distributions. There are many ways to project one distribution over another, but for here, let's say we want to project any arbitrary distribution over exponential family using the KL-*divergence*, or,

$$\hat{q} = \text{proj}\left(p(\mathbf{y}|\mathbf{x}) \to q(\mathbf{y})\right) \triangleq \arg\min_{q} \text{KL}\left(p(\mathbf{y}|\mathbf{x})||q(\mathbf{y})\right)$$

To do so, we choose the following exponential parametric form,

$$q_\theta(\mathbf{y}) = \frac{1}{Z(\theta)} \exp\left(\theta^T \Phi(\mathbf{y})\right), \quad Z(\theta) = \int \exp\left(\theta^T \Phi(\mathbf{y})\right) d\mathbf{y}$$

$\Phi(\mathbf{y})$ is natural statistic of $\mathbf{y}$. The most famous case is a Gaussian distribution with mean and covariance matrix. To reduce the difference between the posterior approximation, $q(.)$ and the real posterior value, again we use the KL-*divergence*.

$$f(\theta) = \text{KL}(p||q) = \mathbb{E}_p \log \frac{p}{q_\theta} = \mathbb{E}_p \log(p) + \mathbb{E}_p \log\left(Z(\theta)\right) - \mathbb{E}_p\left[\theta^T \Phi(\mathbf{y})\right]$$

$$\nabla_\theta f(\theta) = 0 \Rightarrow \nabla_\theta f(\theta) = \nabla_\theta \log Z(\theta) - \mathbb{E}_p[\Phi(\mathbf{y})] = 0 \tag{14.1}$$

We also have:

$$\nabla_\theta \log Z(\theta) = \frac{\nabla_\theta Z(\theta)}{Z(\theta)} = \frac{\nabla_\theta \int \exp\left(\theta^T \Phi(\mathbf{y})\right)}{Z(\theta)} = \frac{\int \nabla_\theta \exp\left(\theta^T \Phi(\mathbf{y})\right)}{Z(\theta)} = \mathbb{E}_q[\Phi(\mathbf{y})] \tag{14.2}$$

Combining the results from equations (14.2, 14.1) we have,

$$\nabla_\theta f(\theta) = 0 \Rightarrow \mathbb{E}_q[\Phi(\mathbf{y})] = \mathbb{E}_p[\Phi(\mathbf{y})] \tag{14.3}$$

Calculating the Hessian for $f(\theta)$ we can show that, the above solution is a *minimum* to $f(\theta)$.

$$[\nabla\nabla_\theta f(\theta)]_{ij} = \frac{\partial^2 \log Z(\theta)}{\partial\theta_j \partial\theta_i} = \frac{\partial}{\partial\theta_j} \frac{\int \Phi_i(\mathbf{y}) \exp\left(\theta^T \Phi(\mathbf{y})\right) d\mathbf{y}}{Z(\theta)} \tag{14.4a}$$

$$= \mathbb{E}_q[\Phi_i(\mathbf{y}), \Phi_j(\mathbf{y})] - \mathbb{E}_q[\Phi_i(\mathbf{y})].\mathbb{E}_q[\Phi_j(\mathbf{y})] \geq 0 \tag{14.4b}$$

Using the above equation, we can conclude that, to get the best estimation for an arbitrary distribution (With KL-*divergence* as the difference between two distributions.) using an exponential distribution, it is enough to match their moments. Specifically if we assume having a Gaussian distribution for the approximating distributions, $q_\theta(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mu, \Sigma)$ Results from equations (14.4, 14.3) are:

$$\begin{cases} \mu^* = \mathbb{E}_p[\mathbf{y}] \\ \Sigma^* = \mathbb{E}_p\left[\mathbf{y}\mathbf{y}^T\right] - \mathbb{E}_p[\mathbf{y}]\mathbb{E}_p[\mathbf{y}]^T \end{cases}$$

14.2.1 ADF for a factorized distribution

Now assume we can factorize the given distribution, $p(\mathbf{y}|\mathbf{x}) = p(\mathbf{x})p(\mathbf{x},\mathbf{y}) = \prod_i t_i(\mathbf{y})$. The goal is to approximate each of the factors using the approximating distribution. It is relatively better to have less components, to keep further calculations minimum, while it is better to have a simple form for each of the factors, so to have easier approximation procedure for each of them. Thus there is a trade-off between the number of factors and simplicity of each factor. The goal is to get the least error by using the minimum number of approximating factors. Assume that the approximating distribution, $q_\theta(\mathbf{y})$ has a known exponential form, e.g. Gaussian. At first we have $q_\theta(\mathbf{y}) = 1$. At each step, we will consider the factor t_i from our target distribution, and changed our approximation based on the added factor t_i to get a better approximation. The distribution $\hat{p}(\mathbf{y})$ is an auxiliary distribution which we use during the algorithm.

$$\hat{p}(\mathbf{y}) = \frac{1}{\tilde{Z}(\theta)} t_i(\mathbf{y}) q_\theta^{old}(\mathbf{y}), \tilde{Z}(\theta) = \int t_i(\mathbf{y}) q_\theta^{old}(\mathbf{y}) d\mathbf{y}. \tag{14.5}$$

$q_\theta^{old}(\mathbf{y})$ is the approximate-posterior distribution at the previous step. Based on the previous definition it is clear that, $\hat{p}(\mathbf{y})$ is the approximate posterior up to factor $t_i(\mathbf{y})$. Similar to what explained, by minimizing $\mathrm{KL}\left(\hat{p}(\mathbf{y}) || q_\theta^{new}(\mathbf{y})\right)$ and assuming that $q_\theta^{new}(\mathbf{y})$ has an exponential form, we use the moments of the distributions to get new approximation, $q_\theta^{new}(\mathbf{y})$. To simplify the notation, we drop *old* from $q_\theta^{old}(\mathbf{y})$. Using the equation (14.5),

$$\nabla_\theta q_\theta(\mathbf{y}) = \nabla_\theta \frac{1}{Z(\theta)} \int \exp\left(\theta^T \Phi(\mathbf{y})\right) d\mathbf{y} = \nabla_\theta \left[\frac{1}{Z(\theta)}\right] \exp\left(\theta^T \Phi(\mathbf{y})\right) + \frac{1}{Z(\theta)} . \nabla_\theta \exp\left(\theta^T \Phi(\mathbf{y})\right)$$

$$\Rightarrow \nabla_\theta q_\theta(\mathbf{y}) = -\frac{\nabla_\theta Z(\theta)}{Z(\theta)} q_\theta(\mathbf{y}) + \Phi(\mathbf{y}) q_\theta(\mathbf{y}) = -\mathbb{E}_q[\Phi(\mathbf{y})] + \Phi(\mathbf{y}) q_\theta(\mathbf{y}).$$

Multiplying in $\frac{1}{\tilde{Z}(\theta)}(\theta).t_i(\mathbf{y})$ and integrating with respect to $\mathbf{y}$ we have,

$$\nabla_\theta \frac{t_i(\mathbf{y})q_\theta(\mathbf{y})}{\tilde{Z}(\theta)} = -\mathbb{E}_q\left[\Phi(\mathbf{y})\right].\frac{1}{\tilde{Z}(\theta)}t_i(\mathbf{y})q_\theta(\mathbf{y}) + \Phi(\mathbf{y}).\frac{1}{\tilde{Z}(\theta)}t_i(\mathbf{y})q_\theta(\mathbf{y}).$$

$$\Rightarrow \frac{1}{\tilde{Z}(\theta)}\nabla_\theta \tilde{Z}(\theta) = -\mathbb{E}_q\left[\Phi(\mathbf{y})\right] + \mathbb{E}_{\hat{p}}\left[\Phi(\mathbf{y})\right].$$

$$\Rightarrow \mathbb{E}_{\hat{p}}\left[\Phi(\mathbf{y})\right] = \nabla_\theta \log\left(\tilde{Z}(\theta)\right) + \mathbb{E}_q\left[\Phi(\mathbf{y})\right].$$

For calculating $q_\theta^{new}(\mathbf{y})$ from $\mathrm{KL}\left(q_\theta^{new}(\mathbf{y})||\hat{p}(\mathbf{y})\right)$ we can use equations (14.4, 14.3) to match the moments for $q_\theta^{new}(\mathbf{y})$ and $\hat{p}(\mathbf{y})$:

$$\mathbb{E}_{q^{new}}\left[\Phi(\mathbf{y})\right] = \mathbb{E}_{\hat{p}}\left[\Phi(\mathbf{y})\right].$$

$$\Rightarrow \mathbb{E}_{q^{new}}\left[\Phi(\mathbf{y})\right] = \nabla_\theta \log\left(\tilde{Z}(\theta)\right) + \mathbb{E}_q\left[\Phi(\mathbf{y})\right].$$

If we assume an exponential distribution we can find $\nabla_\theta \log\left(\tilde{Z}(\theta)\right)$ and $\mathbb{E}_q\left[\Phi(\mathbf{y})\right]$ in closed form. Assume a Gaussian distribution, $q_\theta(\mathbf{y}) = q(\mathbf{y};\mu,\Sigma) = \mathcal{N}(\mathbf{y};\mu,\Sigma)$ and $\tilde{Z}(\theta) = \tilde{Z}(\mu,\Sigma) = \int t(\mathbf{y}).q(\mathbf{y};\mu,\Sigma)\,d\mathbf{y}$. We now have:

$$\nabla_\mu q(\mathbf{y};\mu,\Sigma) = \Sigma^{-1}(\mathbf{y}-\mu)\,q(\mathbf{y};\mu,\Sigma)$$

$$\Rightarrow \mathbf{y}.q(\mathbf{y};\mu,\Sigma) = \mu.q(\mathbf{y};\mu,\Sigma) + \Sigma.\nabla_\mu q(\mathbf{y};\mu,\Sigma)$$

Multiplying both sides in $\frac{1}{\tilde{Z}}t_i(\mathbf{y})$ and integrating with respect to $\mathbf{y}$,

$$\mu^* = \mu + \frac{1}{\tilde{Z}}\Sigma.\nabla_\mu \int t(\mathbf{y})q(\mathbf{y})d\mathbf{y} \tag{14.6a}$$

$$= \mu + \frac{1}{\tilde{Z}}\Sigma.\nabla_\mu \tilde{Z}(\mu,\Sigma) \tag{14.6b}$$

$$= \mu + \Sigma.\nabla_\mu \log\left(\tilde{Z}(\mu,\Sigma)\right) \tag{14.6c}$$

$$= \mu + \Sigma.\mathbf{g}, \qquad \mathbf{g} \triangleq \nabla_\mu \log\left(\tilde{Z}(\mu,\Sigma)\right) \tag{14.6d}$$

Similarly we for covariance (second moment) we have,

$$\Rightarrow \mathbf{y}\mathbf{y}^\top q(\mathbf{y}) = 2\Sigma.\left[\nabla_\Sigma q(\mathbf{y})\right]$$

$$\Rightarrow \langle \mathbf{y}\mathbf{y}^T\rangle = \Sigma + 2\Sigma\mathbf{G}\Sigma\langle\mathbf{y}\rangle_{\hat{p}(\mathbf{y})}\mu^T + \mu\langle\mathbf{y}\rangle^T_{\hat{p}(\mathbf{y})} - \mu\mu^T, \qquad \mathbf{G} = \nabla_\Sigma \log\left(\tilde{Z}(\mu,\Sigma)\right)$$

$$\Rightarrow \Sigma^* = \langle \mathbf{y}\mathbf{y}^T\rangle_{\hat{p}(\mathbf{y})} - \langle\mathbf{y}\rangle_{\hat{p}(\mathbf{y})}\langle\mathbf{y}\rangle^T_{\hat{p}(\mathbf{y})} = \Sigma - \Sigma\left(\mathbf{g}\mathbf{g}^T - 2\mathbf{G}\right)\Sigma.$$

Based the above formula, it should be clear that, the order in which we use the factors $\{t_i\}_i$ to create the approximation, will change the final answer, since to approximate the posterior we go through the factors linearly only once. While Expectation Propagation aims to solve this problem by finding more consistent approximation by passing the factors for several time, so that the approximation is not dependent on the order which we see the factors. One sample difference in approximation of posterior of a toy example is shown in Figure (14.1).

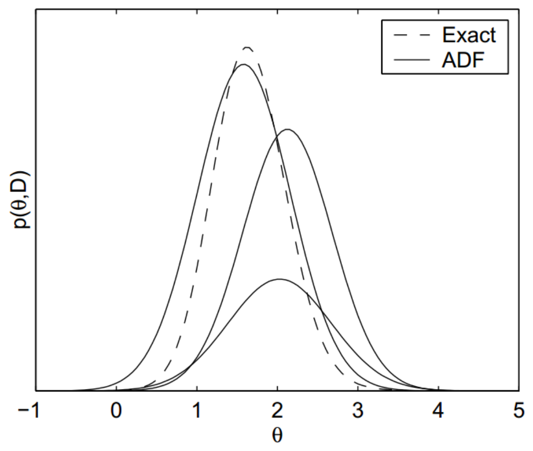


Figure 14.1: The effect of changing the ordering in factors of the target distribution, while approximating via ADF; Minka (2001c).

14.3 Expectation Propagation (EP)

The EP approximation first introduced in Minka (2001c). The method is so much similar to Automatic Density Filtering (ADF). In fact, EP is using update rules of in an intelligent iterative way until convergence. As mentioned the main problem in ADF is that, different order in approximating the factors will result in different approximations. While EP loops over the factors until convergence. Another difference is that, instead of applying the KL-*divergence* to $q_\theta(\mathbf{y})$ as in ADF, in EP we apply it to each factors $t_i(\mathbf{y})$ and then we update the approximation $p(\mathbf{y}|\mathbf{x})$. This way, by sweep over the factors for several time, the ordering of selecting the factors $\{t_i\}_i$ doesn't make any difference. The algorithm is shown in Algorithm (4).

We want to approximate $p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z}\prod_i^n t_i(\mathbf{y})$ using $q_\theta(\mathbf{y}) = \dfrac{\prod_i^n \tilde{t}_i}{\int \prod_i^n \tilde{t}_i}$. We choose the approximating family, $\tilde{t}_i$ to be exponential, to make it easier to work with comparing to t_i itself. Thus we want to approximate any factor t_i with $\tilde{t}_i$, such that the global difference between the exact distribution and the approximating family, $\mathrm{KL}\left(p(\mathbf{y}|\mathbf{x})||q(\mathbf{y})_\theta\right)$ is minimized. It can be shown that the global minimization could be achieved via a set of local minimizations Minka et al. (2005). Note that in this formulation, $\tilde{t}_i$ don't need to be normalized or a proper distribution. For example they can be Gaussians with negative variance (an improper distribution). But given the set of $\{\tilde{t}_i\}_{i=1}^n$ we can normalized their multiplication and get a proper distribution.

Algorithm 4: Expectation Propagation

```
Initialize $\{\tilde{t}_i\}$
$q_\theta(\mathbf{y}) = \frac{\prod_i \tilde{t}_i}{\int \prod_i \tilde{t}_i}$
repeat
    Message elimination: Choose a $\tilde{t}_i$ to do approximation with.
    Remove the factor $\tilde{t}_i$ from approximation, $q_\theta^{-i} = \frac{q_\theta}{\tilde{t}_i}$
    Belief projection: Project the approximate posterior, with $\tilde{t}_i$ replaced with $t_i$, on the
    approximating family,
        $q_\theta^{new}(\mathbf{y}) = \text{proj}\left(\hat{p}_i(\mathbf{y}) \to q_\theta(\mathbf{y})\right),$
    where,
        $\hat{p}_i(\mathbf{y}) = \frac{1}{Z} q_\theta^{-i}(\mathbf{y}) t_i(\mathbf{y}), \ Z = \int q_\theta^{-i}(\mathbf{y}) \times t_i(\mathbf{y}) d\mathbf{y}$
    Message update: Compute the new approximating factor,
        $\tilde{t}_i = Z \frac{q_\theta^{new}(\mathbf{y})}{q_\theta^{-i}(\mathbf{y})}$
until all $\tilde{t}_i$ converge;
```

In the case when the approximating family is from the exponential family, $q_\theta(\mathbf{y}) \propto \exp\left(\eta^\top \phi(\mathbf{y})\right)$, the projection in algorithm 4, $q_\theta^{new}(\mathbf{y}) = \text{proj}\left(\hat{p}_i(\mathbf{y}) \to q_\theta(\mathbf{y})\right)$ is equivalent to the following moment matching,

$$\mathbb{E}_{q_\theta(\mathbf{y})}\left[\phi(\mathbf{y})\right] = \mathbb{E}_{\hat{p}_i(\mathbf{y})}\left[\phi(\mathbf{y})\right] \tag{14.7}$$

Note that this moment matching is distributed over each factor.

To find the marginal likelihood, it is enough to find the following expression:

$$p(\mathbf{x}) \approx \int p(\mathbf{x}|\mathbf{y}) p(\mathbf{y}) d\mathbf{y} = \int \prod_i \tilde{t}_i(\mathbf{y}) d\mathbf{y}.$$

14.4 Problems with the standard EP

EP, though giving nice approximations in many applications is sensitive to outliers, and the cases when the approximating family is not close to the target distribution. This issue is addressed in Qi and Guo (2013) and a relaxed form of EP is introduced which has better convergence properties. Another problem with EP is that, there is no known convergence proof for it.

14.5 EP for inference in graphical models

In a graphical model we can interpret each $\tilde{t}_i$ as an approximate message to the original message t_i. There is a nice discussion of using EP for general factor graphs in Sutton (2005). Minka et al. (2005) also provides a unifying view of different message passing algorithms on graphs.

14.6 EP energy function

There is a discussion of primal/dual energy minimization for EP in Minka (2001a). The primal energy function is,

$$\min_{\hat{p}_i} \max_{q} \left[\sum_i \int_{\mathbf{y}} \hat{p}_i(\mathbf{y}) \log \frac{\hat{p}_i(\mathbf{y})}{t_i(\mathbf{y}) p(\mathbf{y})} d\mathbf{y} - (n-1) \int_{\mathbf{y}} q_\theta(\mathbf{y}) \log \frac{q_\theta(\mathbf{y})}{p(\mathbf{y})} d\mathbf{y} \right] \tag{14.8}$$

with the constraints that

$$\mathbb{E}_{q_\theta(\mathbf{y})}[\phi(\mathbf{y})] = \mathbb{E}_{\hat{p}_i(\mathbf{y})}[\phi(\mathbf{y})], \forall i \qquad \text{(The local moment matching, in Equation 14.7)} \tag{14.9}$$

Using the primal form, one can obtain the dual form, and by setting the gradient of the dual form to zero, one can find the fixed-point updates introduced in the previous sections. In the rest, we show how to obtain this function.

■ **Lemma 14.1 — Variational lower bound on KL divergence.** If $p(.)$ and $q(.)$ are proper distributions defined on $\mathscr{X}$, we can show that,

$$\mathrm{KL}(p||q) = \int_{x \in \mathscr{X}} p(x) \log \frac{p(x)}{q(x)} dx = \max_{\nu} \left[\int_{x \in \mathscr{X}} p(x) \nu(x) dx - \log \int_{x \in \mathscr{X}} q(x) e^{\nu(x)} \right]$$

Proof. Easy enough to take functional derivatives with respect to $\nu(.)$ and observe that $\nu(x) = \log \frac{p(x)}{q(x)} + c$ is indeed a global maximizer for the terms inside the max[.] function. ■

Now we derive the dual EP energy.

■ **Lemma 14.2 — The dual EP energy.** The dual enegy function for EP is,

$$\min_{\nu} \max_{\lambda} \left[(n-1) \log \int_{\mathbf{y}} p(\mathbf{y}) \exp\left(\nu^\top \phi(\mathbf{y})\right) d\mathbf{y} - \sum_{i=1}^{n} \log \int_{\mathbf{y}} \hat{t}_i(\mathbf{y}) p(\mathbf{y}) \exp\left(\lambda_i^\top \phi(\mathbf{y})\right) d\mathbf{y} \right], \tag{14.10}$$

with the constraint that,

$$(n-1)\nu = \sum_i \lambda_i.$$

Proof. The dual can be found by applying Lemma 14.1 for several times. Consider the first part of equation 14.8, which can be lower-bounded using Lemma 14.1.

$$\int_{\mathbf{y}} \hat{p}_i(\mathbf{y}) \log \frac{\hat{p}_i(\mathbf{y})}{t_i(\mathbf{y}) p(\mathbf{y})} d\mathbf{y} = \max_{\lambda} \left[\int_{\mathbf{y}} \hat{p}(\mathbf{y}) \lambda_i(\mathbf{y}) d\mathbf{y} - \log \int_{\mathbf{y}} t_i(\mathbf{y}) p(\mathbf{y}) \exp \lambda_i(\mathbf{y}) d\mathbf{y} \right]$$

Define $\lambda_i(\mathbf{y}) = \lambda_i^\top \phi(\mathbf{y})$, also using the constraint in equation 14.9,

$$\int_{\mathbf{y}} p_i(\mathbf{y}) \log \frac{\hat{p}_i(\mathbf{y})}{t_i(\mathbf{y}) p(\mathbf{y})} d\mathbf{y} = \max_{\lambda} \left[\sum_i \lambda_i \int_{\mathbf{y}} \hat{q}_\theta(\mathbf{y}) \phi_i(\mathbf{y}) d\mathbf{y} - \log \int_{\mathbf{y}} t_i(\mathbf{y}) p(\mathbf{y}) \exp\left(\lambda_i^\top \phi(\mathbf{y})\right) d\mathbf{y} \right],$$

where $\phi_i(\mathbf{y})$ is the i-th element in the $\phi(\mathbf{y})$. Now consider the second part of equation 14.8 which can also be lower-bounded using Lemma 14.1,

$$\int_{\mathbf{y}} q_\theta(\mathbf{y}) \log \frac{q_\theta(\mathbf{y})}{p(\mathbf{y})} d\mathbf{y} = \max_{\nu} \left[\int_{\mathbf{y}} q_\theta(\mathbf{y}) \nu(\mathbf{y}) d\mathbf{y} - \log \int_{\mathbf{y}} p(\mathbf{y}) \exp \nu(\mathbf{y}) d\mathbf{y} \right]$$

$$\Rightarrow -\int_{\mathbf{y}} q_\theta(\mathbf{y}) \log \frac{q_\theta(\mathbf{y})}{p(\mathbf{y})} d\mathbf{y} = \min_{\nu} \left[-\int_{\mathbf{y}} q_\theta(\mathbf{y}) \nu(\mathbf{y}) d\mathbf{y} + \log \int_{\mathbf{y}} p(\mathbf{y}) \exp \nu(\mathbf{y}) d\mathbf{y} \right].$$

Similarly define $\nu_i(\mathbf{y}) = \nu_i{}^\top \phi(\mathbf{y})$,

$$-\int_{\mathbf{y}} q_\theta(\mathbf{y}) \log \frac{q_\theta(\mathbf{y})}{p(\mathbf{y})} d\mathbf{y} = \min_{\nu} \left[-\sum_i \nu_i \int_{\mathbf{y}} q_\theta(\mathbf{y}) \phi_i(\mathbf{y}) d\mathbf{y} + \log \int_{\mathbf{y}} p(\mathbf{y}) \exp \left(\nu^\top \phi(\mathbf{y}) \right) d\mathbf{y} \right].$$

Combining these two results we get,

$$\begin{aligned} \text{Primal} = \max_{\hat{p}_i, q, \lambda} \min_{\nu} \{ \sum_j \sum_i \lambda_i \int_{\mathbf{y}} q_\theta(\mathbf{y}) \phi_i(\mathbf{y}) d\mathbf{y} - \log \int_{\mathbf{y}} t_i(\mathbf{y}) p(\mathbf{y}) \exp \left(\lambda_i{}^\top \phi(\mathbf{y}) \right) d\mathbf{y} \\ - (n-1) \sum_i \nu_i \int_{\mathbf{y}} q_\theta(\mathbf{y}) \phi_i(\mathbf{y}) d\mathbf{y} + (n-1) \log \int_{\mathbf{y}} p(\mathbf{y}) \exp \left(\nu^\top \phi(\mathbf{y}) \right) d\mathbf{y} \}. \end{aligned}$$

By defining $\sum_i \lambda_i = (n-1)\nu$, and dropping $q_\theta(\mathbf{y})$ as a result, we will end up with the dual form in Equation 14.10. ■

Stable fixed points of EP correspond to the *local* minima of its energy function. By taking derivatives of the dual EP energy function, we will end-up with the updates in Algorithm 4. We show this fact in the following.

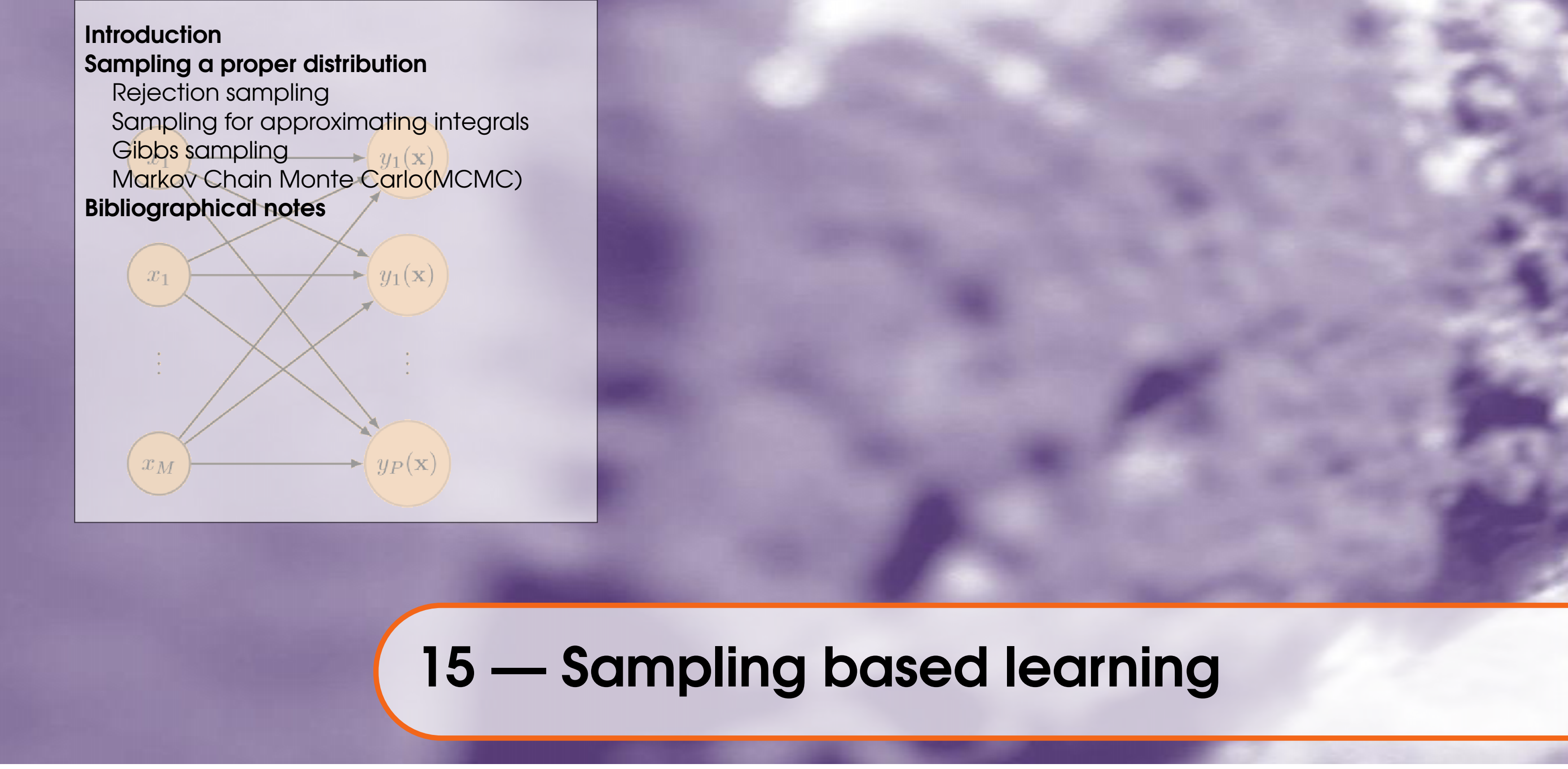
14.6.1 EP updates from dual energy

[TBW]

14.7 Further improvements on EP

Another future work could be in making the inference more efficient. In Seeger and Nickisch (2010) a faster way with convergence guarantees is proposed. Also at Pacheco and Sudderth (2012); Opper et al. (2008) a faster scheme is presented. Minka (2004) is using the α-divergence for mapping the messages. It also showed that, it includes Fractional Belief Propagation Wiegerinck et al. (2003), EP and variational Bayes as special cases. In Kuss and Rasmussen (2005) EP has successfully used for Gaussian Process classification.

14.8 Bibliographical notes



15 — Sampling based learning

15.1 Introduction

In all of the learning problems, after the parametric modelling we need to devise a way to learn the optimal parameters, using some training samplings, or some indirect rules, with respect to some criterion, e.g. a defined loss-function. Usually this can be cast as maximizing (or minimizing, with an additional negative sign) a function of parameters, training data, and the prior knowledge, commonly known as MAP or *maximum a posteriori*.

$$\mathscr{L} = \log p(\mathscr{D}|\Theta) \rightarrow \Theta^* = \max_{\Theta} \log p(\mathscr{D}|\Theta)$$

Since the posterior distribution (or function, if not normalized) is usually a complicated function, it is not straightforward to maximize it directly with respect to model parameters. One approach can be approximating this function and finding the sub-optimal parameters. The other approach which is mostly studied here, is statistical sampling methods, which take many samples of the model, to simulate the behaviour of the model. These methods are usually slow, and exact asymptotically (if they run long enough).

Before starting on learning based on sampling, we should first learn how to sample complicated distributions. Usually it could be assumed that we know how to sample a uniform distribution, and we aim at generalizing it to sampling other complicated distributions.

15.2 Sampling a proper distribution

In theory, there is an easy way to sample any distribution, by finding an invertible parametric form which converts the variables in two distributions. Let's say we know how to sample $p(x)$, our goal is to get the distribution $p(z)$ by finding a parametric form for $x \rightarrow z$. We get the distribution $p(z)$ by the following conversion between two distributions,

$$p(x) = p(z)\left|\frac{dz}{dx}\right|.$$

In Figure 15.1 a probability distribution $p(x)$, and its cumulative distribution $P(X \leq x) =$

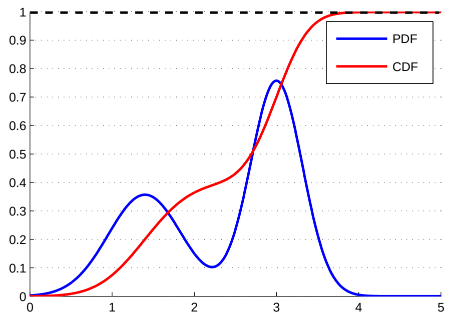


Figure 15.1: A sample distribution, and its cumulative distribution.

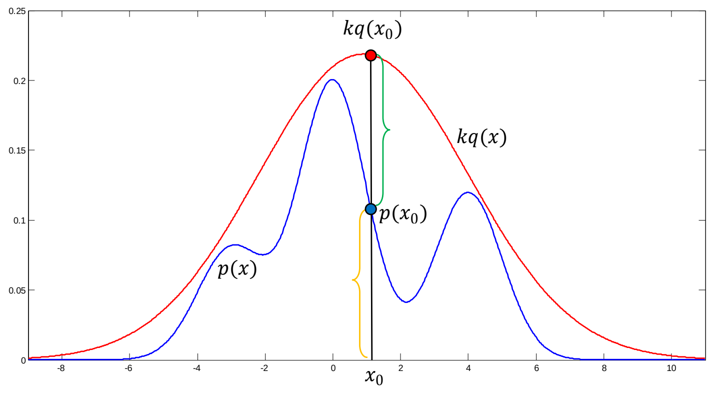


Figure 15.2: A sample distribution, and its cumulative distribution.

$\int_{-\infty}^{x} p(x')dx'$ is depicted. If we sample the y-axis uniformly, find the corresponding points in the CDF curve, and map them on the the x-axis, the corresponding points are distributed according to $p(x)$. In mathematical form, this can be explained in the following form,

$$p(x) = p(z)\left|\frac{dz}{dx}\right|, p(z) = 1 \Rightarrow z = h(x) = P(X \leq x) = \int_{-\infty}^{x} p(x')dx' \Rightarrow x = h^{-1}(z).$$

15.2.1 Rejection sampling

Since in many modelling problems, it is not easy to find an analytical for the CDF, we prefer to find a way for sampling an arbitrary distribution, without the need for finding its CDF. One of these methods is called *rejection sampling*.

Let's say we want to sample a distribution $p(x)$ which has a complicated form, and we can't find its CDF. IJn rejection sampling, we find another distribution $q(x)$ which supports the target distribution, i.e. $p(x)$. In other words, for any x' in the domain of the distributions, $q(x') > p(x')$. Note that, in general $q(x)$ doesn't have to be a proper distribution, in the sense that the area under it sum up to one, but it needs to be of the forms which is easy to sample from. Then $kq(x)$, $k \in \mathbb{R}_{++}$ which is easy to sample from, and supports $p(x)$ can be used. In Figure 15.2 a complicated target distribution$p(x)$, and a supporting distribution $kq(x)$ are shown.

The procedure for rejection sampling is as following: first we sample a point x_0 from the distribution $kq(x)$. Because $k > 0$, we know that $kq(x) > 0$. We create a uniform distributions

on $[0, kq(x)]$, and sample a point from that. If the point is greater that $p(x_0)$ we accept it as a sample of $p(x)$, if not, we reject it. It can be shown that in long-run the accepted samples will have distribution according to $p(x)$ (proof?).
To decrease the ratio of the rejected samples it is necessary to choose the supporting distribution $kq(x)$ as close as possible to $p(x)$, though it might need might be hard to find such a distribution when handling high-dimensional distributions. Also it can be shown, roughly speaking, the probability of a sample being accepted diminishes exponentially with the number of the dimensions. This makes rejection sampling very hard to use in high-dimensional problems, and with a very complicated form, which are hard to visualize. There are a few works which aim at finding better supporting distribution adaptively by using peace-wise exponential functions, or log-concave families (see Gilks and Wild (1992); Gilks et al. (1995))

Usually in probabilistic inference problems, we are dealing with a real ratio of the target distribution $p(x)$. In other words, if we assume that $p(x) = \frac{1}{\mathscr{Z}}\tilde{p}(x)$, where $\mathscr{Z} = \int p(x)dx$ is a normalizing constant, we usually only have $\tilde{p}(x)$, and it is hard to normalize. Thus, there is a big motivation for finding methods which can use the unnormalized function $\tilde{p}(x)$, and give samples of $p(x)$ without directly having it.

15.2.2 Sampling for approximating integrals

Let's say we want to approximate the following integration, $\int f(z)p(z)dz$ which is equivalent to the following expectation, $\mathbb{E}_p[f]$. We can use sample mean as an estimator of the statistical mean, and we can approximate the above expectation by sampling from $p(x)$,

$$\mathbb{E}_p[f] \approx \frac{1}{L}\sum_{i=1}^{L} f(z^i), \quad x^i \sim p(x).$$

Note that in general this trick could be used for approximating any integration with a proper choice of $p(x)$. Also it can easily verified that sample mean is an unbiased estimator the statistical mean.

Let's say we don't know how to sample $p(x)$ and we want to approximate $\mathbb{E}_p[f]$. We choose a distribution $q(x)$ which we know how to sample from, and change the expectation using it,

$$\mathbb{E}_p[f] = \int f(z)p(z)dz = \int f(z)\frac{p(z)}{q(z)}q(z)dz \approx \frac{1}{L}\sum_{i=1}^{L} f(z^i)\frac{p(z^i)}{q(z^i)}, \quad z^i \sim q(z)$$

This let's us sample from $q(x)$ for approximating the value of the sample expectation. This trick is usually called *importance sampling*. Practical usage of this trick demands careful considerations. One important point is that, to get realistic answers, $q(x)$ must be non-zero (or not very small) wherever $p(x)f(x)$ is not zero. This trick has many interesting applications; for example one can use use this trick to calculate expectation of events happening when their probability is very small, e.g. calculating "bit error rate" in a communication system Jeruchim (1984).

Let's consider the case where we don't have the distribution $p(x)$ but we only have a positive ratio of that. In other words, if $p(z) = \frac{1}{Z}\tilde{p}(x)$, we only have $\frac{1}{Z}\tilde{p}(x)$ and calculation of the normalizing constant is too costly that we don't want to do it. We can simplify the previous formulations as following,

$$\mathbb{E}_p[f] = \int f(z)p(z)dz = \frac{1}{Z_p}\int f(z)\tilde{p}(z)dz = \frac{1}{Z_p}\int f(z)\frac{\tilde{p}(z)}{q(z)}q(z)dz = \frac{1}{Z_p}\sum_{i=1}^{L} f(z^i)\frac{\tilde{p}(z^i)}{q(z^i)}, \quad z^i \sim q(z)$$

A similar thing can be done to find an estimation of the normalizing constant,

$$1 = \mathbb{E}_p[1] = \int p(z)dz = \frac{1}{Z_p}\int \tilde{p}(z)dz = \frac{1}{Z_p}\int \frac{\tilde{p}(z)}{q(z)}q(z)dz = \frac{1}{Z_p}\sum_{i=1}^{L}\frac{\tilde{p}(z^i)}{q(z^i)}, \quad z^i \sim q(z)$$

$$\Rightarrow Z_p = \sum_{i=1}^{L}\frac{\tilde{p}(z^i)}{q(z^i)}, \quad z^i \sim q(z)$$

Now using the above unbiased estimations, the estimation for the expectation is as following,

$$\Rightarrow \mathbb{E}_p[f] = \frac{\sum_{i=1}^{L} f(z^i)\frac{\tilde{p}(z^i)}{q(z^i)}}{\sum_{i=1}^{L}\frac{\tilde{p}(z^i)}{q(z^i)}}, \quad z^i \sim q(z).$$

This estimator is *biased* estimator of the target expectation (proof?), it is not always the case that the ratio of any two unbiased estimators is biased estimator (example?).

15.2.3 Gibbs sampling

Let's say we want to sample from a multivariate distribution $p(x,y)$. Since sampling jointly sample from (x,y) we can sample for each variable, from the marginal distributions,

$$\begin{cases} x_t \sim p(x|y_{t-1}) \\ y_t \sim p(y|x_t) \end{cases}$$

In general this can be applied to any distribution with any number of the variables. More details on convergence proof and properties could be found at Smith and Roberts (1993); Raftery and Lewis (1992). The idea of Gibbs sampling in statistics is very similar to “coordinate descent" optimization of multivariate objective functions in optimization(more?).

15.2.4 Markov Chain Monte Carlo(MCMC)

MCMC methods *implicitly* create makov chains which have the stationary distributions the same as that of the target distribution. At each step a new sample $x^{(i)}$ is proposed using a *transition distribution*, $\mathscr{P}(x,x^{'})$,

$$x^{(i-1)} \xrightarrow{\mathscr{P}} x^{(i)}.$$

There are many other names used to call this function, e.g. *Jumping Distribution*, *Proposal Distribution*, *Candidate Generating Distribution*.

One of the families of MCMCs is Metopolis-Hastings methods which introduced in Metropolis et al. (1953); Hastings (1970). Let's say we want to sample from $p(x) = \frac{1}{\mathscr{Z}}\tilde{p}(x)$, and let's assume that we don't have the normalization constant $\mathscr{Z}$. We define a transition distribution,

$$q(z_1,z_2) = \Pr(z_1 \to z_2).$$

The steps of the algorithm are shown in Algorithm 5.

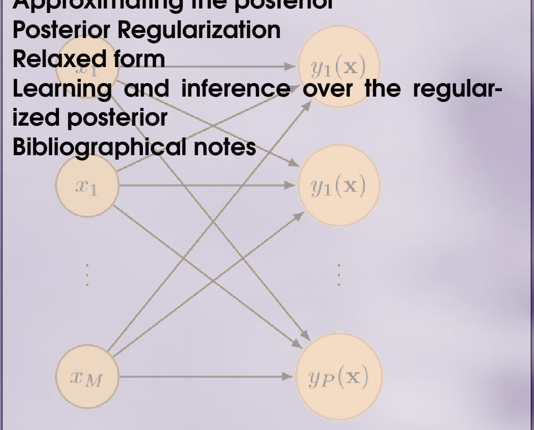
Algorithm 5: Metropolis-Hastings algorithm

```
Start with random samples $z_0$, s.t. $p(z_0) > 0$. **repeat**
    1 genrate random sample, $z^*$ from proposal distribution, $z_* \sim q(Z,z_t)$, given the
    sandom sample of the previous iteration $z_t$.
    Calculate: $\alpha = \min\left\{1, \frac{\tilde{p}(z_*)q(z_*,q_{t-1})}{\tilde{p}(z_{t-1})q(z_{t-1},q_*)}\right\}$.
    $\alpha = \begin{cases} \geq 1 & : \text{Accept the sample: } x_t = z_* \\ < 1 & : \text{Accept the sample with probability of } \alpha. \end{cases}$
**until** *TERMINATION-CONDITION*;
```

To get a good approximation of the samples found from the above method, it is necessary to throw away the samples until a time *burn-in* period k where the samples $x_{k+1}, x_{k+2}, \dots$ get closer to realistic samples of the target distribution, or the markov chain gets close enough to its stationary distribution. More proofs on the convergence and properties could be found at Hastings (1970).

15.3 Bibliographical notes

In preparation of this document I have used Bishop (2006).



16 — Posterior Regularization

16.1 Introduction

One of the key challenges in probabilistic structured learning, is the intractability of the posterior distribution, for fast inference. There are numerous methods proposed to approximate the posterior, so as to make it easy to work with. Posterior Regularization is one of the proposed methods to approximate joint distribution between a set of structured variables, and taking the constrains into account. In Ganchev et al. (2010) there is a comprehensive review of the method and previous ideas is mentioned.

16.2 Modelling the problem

In Posterior Regularization, we are dealing with a doing inference over posterior probability, with considering constraints, as indirect supervision. In this context we define $\mathbf{X}$ as input observation variables, and $\mathbf{Y}$ as latent variables, which we want to make predictions about. One example problem could be POS tagging as an example of structured prediction, in which we see input sentences $\mathbf{x}$, and output tags $\mathbf{y}$. We assume having generative model defined using $p(\mathbf{Y})$ as prior knowledge on latent variables, likelihood to be $p(\mathbf{X}|\mathbf{Y})$, and marginal-likelihood or evidence to be $\mathscr{L}(\boldsymbol{\theta}) = \ln p(\mathbf{X}; \boldsymbol{\theta}) = \ln \int_{\mathbf{Y} \in \mathscr{Y}} p(\mathbf{Y}) p(\mathbf{X}|\mathbf{Y}) d\mathbf{Y}$. However one can use this probabilistic modelling to learn in discriminative way.

16.3 Approximating the posterior

Now we want to use $q(\mathbf{Y})$ to approximate the real intractable posterior, $p(\mathbf{Y}|\mathbf{X};\boldsymbol{\theta})$. In fact, $q(\mathbf{Y})$ is a simpler parametric distribution with parameter γ which is easy to work with [1]. To approximate the true posterior $p(.)$ using the decomposed distribution $q(.)$ we should find a measure of distance between two functions, and also is practical in computational sense. One of the famous distance measure between functions is called Kullback-Leibler divergence or in short, KL-*divergence*.

[1]Thus the right way is to denote it by $q(\mathbf{Y};\gamma)$, but it is common to drop the parameters.

■ **Lemma 16.1** We have

$$\mathscr{L}(\theta) = \log p(\mathbf{X};\theta) = \mathscr{J}(q,\theta) + \mathrm{KL}\left(q(\mathbf{Y})||p(\mathbf{Y}|\mathbf{X};\theta)\right). \tag{16.1}$$

in which

$$\mathscr{J}(q,\theta) \triangleq \int q(\mathbf{Y}) \log \frac{p(\mathbf{X},\mathbf{Y};\theta)}{q(\mathbf{Y})} d\mathbf{Y} \tag{16.2}$$

$$\mathrm{KL}\left(q(\mathbf{Y})||p(\mathbf{Y}|\mathbf{X};\theta)\right) \triangleq \mathbb{E}_q \left[\log \frac{q(\mathbf{Y})}{p(\mathbf{Y}|\mathbf{X};\theta)}\right]$$

Proof. Now the Bayes rule we have,

$$\log p(\mathbf{X};\theta) = \log p(\mathbf{X},\mathbf{Y};\theta) - \log p(\mathbf{Y}|\mathbf{X};\theta)$$
$$\log p(\mathbf{X};\theta) = \log \frac{p(\mathbf{X},\mathbf{Y};\theta)}{q(\mathbf{Y})} - \log \frac{p(\mathbf{Y}|\mathbf{X};\theta)}{q(\mathbf{Y})}.$$

Multiplying two sides in $q(\mathbf{Y})$ we have,

$$q(\mathbf{Y})\log p(\mathbf{X};\theta) = q(\mathbf{Y})\log \frac{p(\mathbf{X},\mathbf{Y};\theta)}{q(\mathbf{Y})} - q(\mathbf{Y})\log \frac{p(\mathbf{Y}|\mathbf{X};\theta)}{q(\mathbf{Y})}$$
$$\int q(\mathbf{Y})\log p(\mathbf{X};\theta)d\mathbf{Y} = \int q(\mathbf{Y})\log \frac{p(\mathbf{X},\mathbf{Y};\theta)}{q(\mathbf{Y})}d\mathbf{Y} - \int q(\mathbf{Y})\log \frac{p(\mathbf{Y}|\mathbf{X};\theta)}{q(\mathbf{Y})}d\mathbf{Y}.$$

Note that in the left part of the above equation $p_{\mathbf{X}}(\mathbf{x})$ is not a function of $\mathbf{Y}$ and thus, $\int q(\mathbf{Y})\log p_{\mathbf{X}}(\mathbf{X}|\theta)d\mathbf{Y} = \log p(\mathbf{X}|\theta)$. Also note that,

$$\begin{aligned}
\mathrm{KL}\left(q(\mathbf{Y})||p(\mathbf{Y}|\mathbf{X};\theta)\right) &\triangleq -\int q(\mathbf{Y})\log \frac{p(\mathbf{Y}|\mathbf{X};\theta)}{q(\mathbf{Y})}d\mathbf{Y} \\
&= \int q(\mathbf{Y})\log \frac{q(\mathbf{Y})}{p(\mathbf{Y}|\mathbf{X};\theta)}d\mathbf{Y} \\
&= \mathbb{E}_q\left[\log \frac{q(\mathbf{Y})}{p(\mathbf{Y}|\mathbf{X};\theta)}\right] \\
&\Rightarrow \mathscr{L}(\theta) = \mathscr{J}(q,\theta) + \mathrm{KL}\left(q(\mathbf{Y})||p(\mathbf{Y}|\mathbf{X};\theta)\right).
\end{aligned}$$

■

■ **Lemma 16.2** For any given observation and latent variables we have the following inequality:

$$\mathscr{L}(\theta) \geq \mathscr{J}(q,\theta)$$

Proof. I use Jensen's inequality here to show that $\mathscr{J}(q,\theta)$ a lower bound for the original likelihood:

$$\begin{aligned}
\mathscr{L}(\theta) = \ln p(\mathbf{X};\theta) = \ln \int p(\mathbf{X},\mathbf{Y};\theta)d\mathbf{Y} \\
= \ln \int q(\mathbf{Y})\frac{p(\mathbf{X},\mathbf{Y};\theta)}{q(\mathbf{Y})}d\mathbf{Y} \\
\geq \int q(\mathbf{Y})\ln \frac{p(\mathbf{X},\mathbf{Y};\theta)}{q(\mathbf{Y})}d\mathbf{Y} = \mathscr{J}(q,\theta)
\end{aligned}$$

This shows that $\exp[\mathscr{J}(q,\theta)]$ a lower bound for likelihood $p(\mathbf{X};\theta)$:

$$\Rightarrow p(\mathbf{X}|\theta) \geq \exp[\mathscr{J}(q,\theta)] \text{ or } \mathscr{L}(\theta) \geq \mathscr{J}(q,\theta).$$

■

Corollary 16.1 Maximizing $\mathscr{J}(q,\theta)$ (lower bound) will result in maximizing the likelihood $p(\mathbf{X};\theta)$. For this reason, $\mathscr{J}(q,\theta)$ is sometimes called **ELBO** (Evidence Lower Bound).

16.4 Posterior Regularization

We define the following to be the objective function to be maximized:

$$\begin{cases} \mathscr{J}(\theta,q) = \mathscr{L}(\theta) - \mathrm{KL}\left(q(\mathbf{Y})||p(\mathbf{Y}|\mathbf{X};\theta)\right) \\ \mathbb{E}_q\left[\phi(\mathbf{X},\mathbf{Y})\right] \leq \mathbf{b} \end{cases} \tag{16.3}$$

Observation 1. *Having the above inequality lets us to define the the equation (16.1), since KL-divergence is always a positive value.*

We define a set of constraints in the output space as following:

$$\mathscr{Q} = \left\{q(\mathbf{Y})|\mathbb{E}_q\left[\phi(\mathbf{X},\mathbf{Y})\right] \leq \mathbf{b}\right\}$$

Note that in the definition of the constraints $\mathbb{E}_q\left[\phi(\mathbf{X},\mathbf{Y})\right]$ is a function of $\mathbf{X}$ which means that we define our constraints over input observations, to follow the structure defined by inequality and in the feature function $\phi(\mathbf{X},\mathbf{Y})$. This definition can encode hard and soft constraints in itself.

16.5 Relaxed form

To cut some slack on objective function and the constraint function, one could define extra slack variables:

$$\begin{cases} \mathscr{J}(\theta,q) = \mathscr{L}(\theta) - \mathrm{KL}\left(q(\mathbf{Y})||p(\mathbf{Y}|\mathbf{X};\theta)\right) + \sigma||\xi||_\beta \\ \mathbb{E}_q\left[\phi(\mathbf{X},\mathbf{Y})\right] - \mathbf{b} \leq \xi \end{cases}$$

16.6 Learning and inference over the regularized posterior

Without considering the constraints, we can show that training consists of a two step procedure similar to EM algorithm:

1. Initialize θ, and variational parameters of $q(.)$.
2. Repeat until convergence
   (a) E-step: $q^{(t+1)} = \arg\max_q \mathscr{J}(q^{(t)},\theta^{(t)})$
   (b) M-step: $\theta^{(t+1)} = \arg\max_\theta \mathscr{J}(q^{(t+1)},\theta^{(t)})$

It is worthy to mention that minimizing the objective function is equivalent to maximizing the KL-*divergence*.

By considering the constraint, the above procedure is limited to those $q(.)$s that satisfy the constraint set $\mathscr{Q}$. One way to do so, it to constrain the “E-step" to $\mathscr{Q}$:

$$\text{E-step(modifiede): } q^{(t+1)} = \arg\max_{q\in\mathscr{Q}} \mathscr{J}(q^{(t)},\theta^{(t)})$$

Instead of applying such a constraint which is hard to calculate, we can use the dual form of the “E-step" and the constraint inequality.

Theorem 16.1 Consider the following problem:

$$\begin{cases} \max_{q,\xi} \text{KL}\left(q(\mathbf{Y})||p(\mathbf{Y}|\mathbf{X};\theta)\right) \\ \mathbb{E}_q\left[\phi(\mathbf{X},\mathbf{Y})\right] - \mathbf{b} \leq \xi \end{cases} \tag{16.4}$$

With θ as constant parameters. The primal solution $q^*(\mathbf{Y})$ is unique since KL-*divergence* is strictly convex:

$$q^*(\mathbf{Y}) = \frac{p(\mathbf{Y}|\mathbf{X};\theta)\exp\left\{-\lambda^*.\phi(\mathbf{X},\mathbf{Y})\right\}}{Z(\lambda^*)}$$

in which $Z(\lambda^*) = \int p(\mathbf{Y}|\mathbf{X};\theta)\exp\left\{-\lambda^*.\phi(\mathbf{X},\mathbf{Y})\right\}d\mathbf{Y}$ is the normalizing function. The dual for the above program is as following:

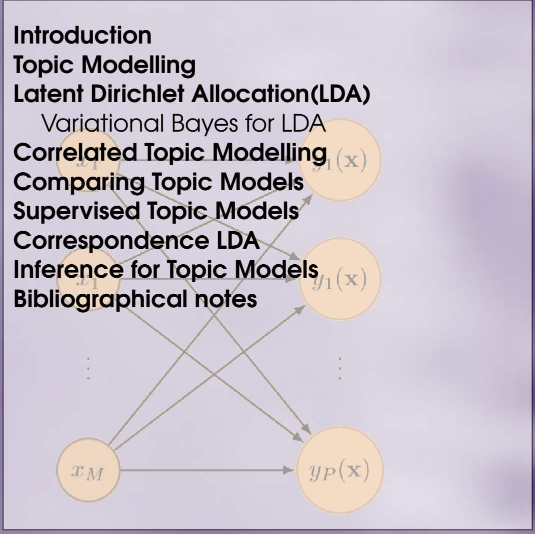
$$\max_{\lambda\geq 0}\left[-\mathbf{b}.\lambda - \ln Z(\lambda) - \varepsilon\|\lambda\|_{\beta'}\right]$$

in which $\|.\|_\beta$ is dual norm for $\|.\|_{\beta'}$.

The proof can be found at Ganchev et al. (2010).

16.7 Bibliographical notes

I used Jiayu Zhou's notes (`http://www.public.asu.edu/~jzhou29/slides/PR_Notes.pdf`) in writing this document.



17 — Topic Modelling

17.1 Introduction

[TBW]

17.2 Topic Modelling

Since the introduction of Latent Dirichlet AllocationBlei et al. (2003), for latent clustering of documents, due to the flexibility of the mode, there has been numerous works on extending this work to many interesting problems, with different models, and inference methods. These applications ranged from many NLP applications, like text-author modelling, to many other applications, like in computer vision, for modelling image clustering.

Though very flexible, and easy to work in nature, for many applications, the major problem is the inclusion of semantic informations into the problem; while in many cases, one could exploit the structural differences to create clustering, there are many cases that are not easy to capture, and the differences lie in semantics. Thus, there has been great deal of efforts to make the clustering in LDA-like models more coherent and meaningful.

In topic models the goal is to develop tools for statistical analysis of document collections. There has been a lot of works mainly initiated by Blei et al. (2003); let's say we have a bunch of documents, each of which have bunch of words. The connection between these documents has some interesting properties. To create a model that could capture the mutual connection between the documents, it is assumed that set of latent variables Z which is set of topics. Each document is comprised of several topics, with some proportions. Some documents might share topics. Two documents are semantically more closer to each other if they share more common topics with similar proportions. To find the topic proportion of each document, one needs to look into the contents of the documents, i.e. words. Thus for each word there is a probability of membership to each topic. To model this, we consider a multinomial θ_i representing topic distribution for the i-th document and a multinomial ϕ_k representing the word distribution for topic k. To make the model robust to overfitting, we put Dirichlet priors on each of the variables. Now the model could be learnt using mean-field variational approximation of the likelihood.

This document includes a comprehensive review of the related works, and their properties.

In Section 2, we are summarizing the LDA-like models for relation extraction. In section, I am reviewing semi-supervision in topic modelling. Secion 3, reviews some of the related applications of topic modelling. Section 4 is about making topic modelling more semantically meaningful and coherent. And Section 5 explains the proposed model and the progress in that model.

17.3 Latent Dirichlet Allocation(LDA)

In topic models [1] the goal is to develop tools for statistical analysis of a set of documents. There has been a lot of works mainly initiated by Blei et al. (2003) and some related works on probabilistic document modelling specially Latent Semantic Indexing (LSI) Hofmann (1999). The graphical model for the model is shown in Figure 17.1 and the parameters are in Table 17.1. Let's say we have D documents, each of them comprised of N words. The connection between these documents has some interesting properties. To create a model that could capture the mutual connection between the documents, it is assumed that set of latent variables Z which acts like a switch which assigns topics to each words. Each documents is comprised of several topics, with some proportions. Some documents might share topics. Two documents are semantically more closer to each other if they share more common topics with similar proportion. To find the topic proportion of each document, one needs to look into the contents of the documents, i.e. words. Thus for each word there is a probability of membership to each topic. To model this, we consider a multinomial θ_i representing topic distribution for the i-th document and a multinomial ϕ_k representing the word distribution for topic k. To make the model robust to over-fitting, we put Dirichlet priors on each of the multinomials.

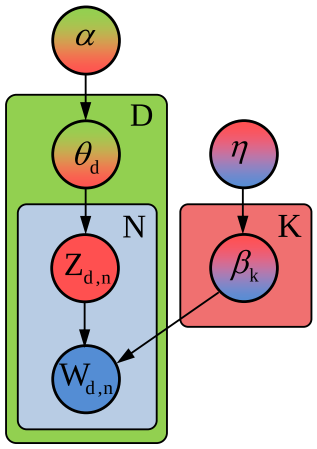


Figure 17.1: Graphical model for Latent Dirichlet Allocation.

As it could be seen from the model, each topic is defined over set of words. The prior over the distribution of topic in documents, is shared over all of the documents. This imposes a similarity of topic distribution among documents, but not restricted. Now we could sample this generative model (generate set of documents, topics for each document, and words generated based on the topic distributions) as follows

1. For each topic k,
   (a) Select a random topic proportion over words , $\beta_k \sim \text{Dir}(\eta)$.
2. For each document d,
   (a) Select a random topic proportion per documents, $\theta_d \sim \text{Dir}(\alpha)$
   (b) For each word,

[1] To make explanations more clear I am using green to denote document, red for topic, and blue for word.

D	# of documents.
N	# of words.
K	# of topics.
V	size of vocabulary.
α	A positive K-vector, topic/document Dirichlet parameter.
η	A scalar positive value, word/topic Dirichlet parameter(a symmetric distribution).
θ_d	distribution of topics for the i-th document, $\theta_d \sim \text{Dir}(\alpha)$.
β_k	distribution of words given k-th topic, $\beta_k \sim \text{Dir}(\eta)$.
$Z_{d,n}$	topic assignment of words, e.g. if $Z_{i,j} = k$, the j-th word of i-th document has k-th topic. specifically we have $Z_{d,n} \sim \text{Mult}(\theta_d), Z_{d,n} \in \{1,\dots,K\}$.
$W_{d,n}$	generated word, for example $W_{i,j}$ is the j -th word from the i-th document. specifically $W_{d,n} \sim \text{Mult}(\beta_{Z_{d,n}}), W_{d,n} \in \{1,\dots,V\}$

Table 17.1: Parameters of LDA.

i. Select a random topic, $Z_{d,n} \sim \text{Mult}(\theta_d), Z_{d,n} \in \{1,\dots,K\}$.
ii. Select a random word, $W_{d,n} \sim \text{Mult}(\beta_{Z_{d,n}}), W_{d,n} \in \{1,\dots,V\}$

Now having the model given, we can write the posterior over variables, given observations and hyper-parameters, could be written as following:

$$p(\theta_{1:D}, Z_{1:D,1:N}, \beta_{1:K} | W_{1:D,1:N}, \alpha, \eta) = \frac{p(\theta_{1:D}, Z_{1:D}, \beta_{1:K} | W_{1:D}, \alpha, \eta)}{\int_{\theta_{1:D}} \int_{\beta_{1:K}} \sum_{Z_{1:D}} p(\theta_{1:D}, Z_{1:D}, \beta_{1:K} | W_{1:D}, \alpha, \eta)} \tag{17.1}$$

The complex integration+summation at the denominator of the above formula makes it intractable to directly use the above formula for estimation of model parameters. Thus we try to devise other tricks to solve this problem.

17.3.1 Variational Bayes for LDA

In variational inference, we approximate the posterior over variables inside model, by assuming a set of independence assumptions, and decompose the posterior into smaller distributions. One decomposition for the variables inside our model could be as follows:

$$q(\theta_{1:D}, z_{1:D,1:N}, \beta_{1:K}) = \prod_{k=1:K} q(\beta_k | \lambda_k) \prod_{d=1:D} \left\{ q(\theta_d | \gamma_d) \prod_{n=1:N} q(z_{d,n} | \phi_{d,n}) \right\}$$

This decomposition decouples the dependence between variables; the decomposed graphical model is shown in Figure 17.2. We use the decomposed version of the posterior and minimize its and the original posterior's difference, to find the variational parameters of the approximated distribution:

$$(\lambda^*_{1:K}, \gamma^*_{1:D}, \phi^*_{1:D,1:N}) = \\ \arg \min_{\lambda_{1:K}, \gamma_{1:D}, \phi_{1:D,1:N}} \text{KL}\left(q(\theta_{1:D}, z_{1:D,1:N}, \beta_{1:K}) || p(\theta_{1:D}, z_{1:D,1:N}, \beta_{1:K} | w_{1:D,1:N}, \alpha, \eta)\right)$$

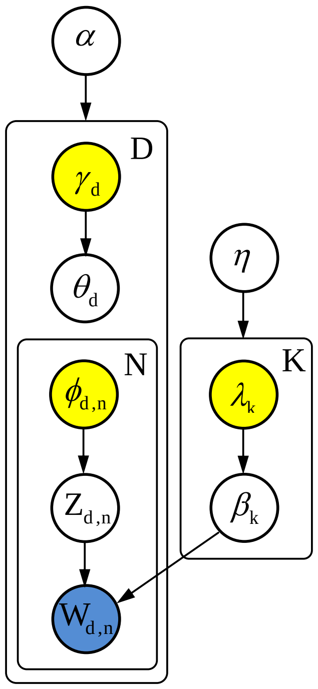


Figure 17.2: Decomposition of LDA; the newly added parameters are shown in yellow. The blue variable is observation.

For simplicity we drop the index of variables.

$$\begin{aligned}
\mathscr{L} = \ln p(w|\alpha,\eta) &= \ln \int_\theta \int_\beta \sum_z p(w,\theta,\beta,z|\alpha,\eta) \mathrm{d}\theta \mathrm{d}\beta \\
&= \ln \int_\theta \int_\beta \sum_z q(\theta,\beta,z|\gamma,\phi,\lambda) \frac{p(w,\theta,\beta,z|\alpha,\eta)}{q(\theta,\beta,z|\gamma,\phi,\lambda)} \mathrm{d}\theta \mathrm{d}\beta \\
&\geq \int_\theta \int_\beta \sum_z q(\theta,\beta,z|\gamma,\phi,\lambda) \ln \frac{p(w,\theta,\beta,z|\alpha,\eta)}{q(\theta,\beta,z|\gamma,\phi,\lambda)} \mathrm{d}\theta \mathrm{d}\beta \\
&= \mathbb{E}_q \left[\ln \frac{p(w,\theta,\beta,z|\alpha,\eta)}{q(\theta,\beta,z|\gamma,\phi,\lambda)} \right] = \mathscr{L}(\gamma,\phi,\lambda;\alpha,\eta)
\end{aligned}$$

$$\begin{aligned}
\mathscr{L}(\gamma,\phi,\lambda;\alpha,\eta) = & \underbrace{\mathbb{E}_q \left[\sum_d^D \ln p(\theta_d|\alpha) \right]}_{\text{T1}} + \underbrace{\mathbb{E}_q \left[\sum_d^D \sum_n^N \ln p(z_{d,n}|\theta_d) \right]}_{\text{T2}} + \underbrace{\mathbb{E}_q \left[\sum_k^K \ln p(\beta_k|\eta) \right]}_{\text{T3}} + \\
& + \underbrace{\mathbb{E}_q \left[\sum_d^D \sum_n^N \ln p(w_{d,n}|z_{d,n},\beta_{z_{d,n}}) \right]}_{\text{T4}} - \underbrace{\mathbb{E}_q \left[\sum_d^D \ln q(\theta_d|\gamma_d) \right]}_{\text{T5}} - \underbrace{\mathbb{E}_q \left[\sum_k^K \ln q(\beta_k|\lambda_k) \right]}_{\text{T6}} \\
& - \underbrace{\mathbb{E}_q \left[\sum_d^D \sum_n^N \ln q(z_{d,n}|\phi_{d,n}) \right]}_{\text{T7}}
\end{aligned}$$

For a Dirichlet distribution, we can find an analytic expression for the expectation of the logarithm of its variable:

$$\mathbb{E}_{\theta\sim\text{Dir}(\alpha)}[\ln\theta_k] = \psi(\alpha_k) - \psi(\sum_i \alpha_i)$$

Where $\psi(x)$ is *digamma* function,

$$\psi(x) = \frac{d}{dx}\ln\Gamma(x) = \frac{\Gamma'(x)}{\Gamma(x)}$$

And similarly

$$\mathbb{E}_{\theta\sim\text{Dir}(\alpha)}[\ln p(\theta|\alpha)] = \ln\Gamma(\sum_i \alpha_i) - \sum_i \ln\Gamma(\alpha_i) + \sum_i (\alpha_i - 1)(\psi(\alpha_i) - \psi(\sum_{\hat{i}} \alpha_{\hat{i}}))$$

Using the above identities we simplify each of the terms in the lower-bound of the likelihood:

$$\text{T1} = \sum_d^D \left[\ln\Gamma(\sum_k^K \alpha_k) - \sum_k^K \ln\Gamma(\alpha_k) + \sum_k^K (\alpha_k - 1)(\psi(\gamma_{d,k}) - \psi(\sum_{k'}^K \gamma_{d,k'}))\right]$$

$$\text{T2} = \sum_d^D \sum_n^N \sum_k^K \phi_{d,n,k}(\psi(\gamma_{d,k}) - \psi(\sum_{k'} \gamma_{d,k'}))$$

$$\text{T3} = \sum_k^K \left[\ln\Gamma(\sum_v^V \eta_v) - \sum_v^V \ln\Gamma(\eta_v) + \sum_v^V (\eta_v - 1)(\psi(\lambda_{z,v}) - \psi(\sum_{v'}^V \lambda_{z,v'}))\right]$$

$$\text{T4} = \sum_d^D \sum_n^N \sum_k^K \phi_{d,n,k}(\psi(\lambda_{k,w_{d,n}}) - \psi(\sum_{v'}^V \lambda_{k,v'}))$$

$$\text{T5} = \sum_d^D \left[\ln\Gamma(\sum_k^K \gamma_{d,k}) - \sum_k^K \ln\Gamma(\gamma_{d,k}) + \sum_k^K (\gamma_{d,k} - 1)(\psi(\gamma_{d,k}) - \psi(\sum_{k'}^K \gamma_{d,k'}))\right]$$

$$\text{T6} = \sum_k^K \left[\ln\Gamma(\sum_v^V \lambda_{k,v}) - \sum_v^V \ln\Gamma(\lambda_{k,v}) + \sum_v^V (\lambda_{k,v} - 1)(\psi(\lambda_{k,v}) - \psi(\sum_{v'}^V \lambda_{k,v'}))\right]$$

$$\text{T7} = \sum_d^D \sum_n^N \sum_k^K \phi_{d,n,k}\ln\phi_{d,n,k}$$

E-step: (equivalent to minimizing KL(.||.))

$$\frac{\partial\mathscr{L}}{\gamma_{d*,k*}} = 0 \Rightarrow \gamma_{d*,k*} = \alpha_{k*} + \sum_n^N \phi_{d*,n,k*}$$

$$\frac{\partial\mathscr{L}}{\partial\lambda_{k*,v*}} = 0 \Rightarrow \lambda_{k*,v*} = \eta_{v*} + \sum_d^D \sum_n^N \phi_{d,n,k*}\mathrm{I}_{w_{d,n}=v*}$$

$$\frac{\partial}{\partial\phi_{d*,n*,k*}}\{\mathscr{L} - \lambda_{d*,n*}(\sum_k \phi_{d*,n*,k} - 1)\} = 0 \Rightarrow$$

$$\phi_{d*,n*,k*} \propto \exp\{\psi(\gamma_{d*,k*}) - \psi(\sum_{k'} \gamma_{d*,k'}) + \psi(\lambda_{k*,w_{d*,n*}}) - \psi(\sum_{v'}^V \lambda_{k*,v'})\}$$

M-step: Updates for the gamma distribution. The updating could be done using Newton-Raphson optimization:

$$\frac{\partial \mathscr{L}}{\partial \alpha_{k*}} = \sum_{d}^{D}[\psi(\sum_{k}^{K}\alpha_k) - \psi(\alpha_{k*}) + \psi(\gamma_{d,k*}) - \psi(\sum_{k'}^{K}\gamma_{d,k'})]$$

$$\frac{\partial^2 \mathscr{L}}{\partial \alpha_{k_1}\partial \alpha_{k_2}} = \sum_{d}^{D}[\psi'(\sum_{k}^{K}\alpha_k) - \psi'(\alpha_{k_1})\mathrm{I}_{k_1=k_2}]$$

Inference using Gibbs sampling

Gibbs sampling is considered as a special variant of Metropolis-Hastings algorithm, also a Monte-Carlo Markov Chain (MCMC) sampling method. We can show that,

$$p(z_i = j|\mathbf{z}_{-i}, \mathbf{w}, \alpha, \eta) \propto p(z_i = j, \mathbf{z}_{-i}, \mathbf{w}|\alpha, \eta) = p(\mathbf{z}, \mathbf{w}|\alpha, \eta)$$

The above distribution also shows the exchangeablity property for each of the random variables in $\mathbf{z}$. Now we can simplify $p(z_i = j|\mathbf{z}_{-i}, \mathbf{w}, \alpha, \eta)$ based on the words counts for topics and documents. To do so, we can expand the above distribution,

$$\begin{aligned} p(\mathbf{z}, \mathbf{w}, \alpha, \eta) &= \int\int p(\mathbf{z}, \mathbf{w}, \theta, \beta|\alpha, \eta)d\theta d\beta \\ &= \int\int p(\mathbf{z}|\theta)p(\mathbf{w}|\beta)p(\theta|\alpha)p(\beta|\eta)d\theta d\beta \\ &= \int p(\mathbf{z}|\theta)p(\theta|\alpha)d\theta \int p(\mathbf{w}|\beta)p(\beta|\eta)d\beta \\ &= \prod_d \frac{B(n^d_{.,j}+\alpha)}{B(n^d_{-i,j}+\alpha)} \prod_w \frac{B(n^w_{.,j}+\eta)}{B(n^w_{-i,j}+\eta)} \end{aligned}$$

Using a few simplification we could show that,

$$p(z_i = j|\mathbf{z}_{-i}, \mathbf{w}, \alpha, \eta) = \underbrace{\frac{n^{w_i}_{-i,j}+\eta}{\sum_{w_i} n^{w_i}_{-i,j} + W\eta}}_{\text{Probability of } w_i \text{ under topic } j} \overbrace{\frac{n^{d_i}_{-i,j}+\alpha}{\sum_{d_i} n^{d_i}_{-i,j} + T\alpha}}^{\text{Probability of } z_i \text{ in document containing } w_i} \quad (17.2)$$

Using the above probability we can do the sampling as follows

Comparing variational procedure with Gibbs sampling, Variational procedure is faster, but Gibbs sampling is guaranteed to find the global optimum answer, and if used appropriately it is more accurate. Is also has an easy setup. But the downside with the Gibbs sampling is that, there is no concrete results on its convergence rate. In practice it might take infinite long time.

17.4 Correlated Topic Modelling

One deficiency in the previous model defined for topic modelling, it the independence assumption between topics. The independence assumption comes from using Dirichlet prior over topics, in which the correlation between topics is not taken into account. In Blei and Lafferty (2006) the *Correlated Topic Models* is suggested in which, it replaces Dirichlet with *Logistic Normal Distribution*, which is also a distribution over a simplex for a richer class of distributions, and unlike Dirichlet captures better inter-component correlations Huang and Malisiewicz (2009). Now it just needs to train this model similar to LDA which are extensively discussed in Blei and Lafferty (2006).

Algorithm 6: The sampling procedure for LDA

```
Data: documents and words.
Result: topic assignments $Z_{d,n}$
Randomly initialize the topic assignments $Z_{d,n}$;
while not-converged do
    for each document, $d \in \{1, \ldots, D\}$ do
        for each word, $n, \in \{1, \ldots, N\}$ do
            $w \leftarrow W_{d,n}$
            $z \leftarrow Z_{d,n}$
            $n_{d,z} \leftarrow 1; n_{w,z} \leftarrow 1; n_z \leftarrow 1$
            for each topic, $K \in \{1, \ldots, K\}$ do
                $p(z_i = j|\mathbf{z}_{-i}, \mathbf{w})$ using equation 17.2
            end
            $Z_{d,n} \leftarrow$ Sample( $p(z_i = j|\mathbf{z}_{-i}, \mathbf{w})$ )
            $z \leftarrow Z_{d,n}$
            $n_{d,z} \leftarrow n_{d,z} + 1; n_{w,z} \leftarrow n_{w,z} + 1; n_z \leftarrow n_z + 1$
        end
    end
end
```

17.5 Comparing Topic Models

In Boyd-Graber et al. (2009) it is trying to do so, i.e. bring a good interpretation and causality behind latent variables. In this paer, using human experiments, tries to analyze the performance of each of the models for LDA, CTM and pLSI, relevance of topic models. The first is *word intrusion* which “measures how semantically cohesive the topics inferred by a model"are" and “tests whether topics correspond to natural groupings for humans". The second one is *topic intrusion* is measures “how well a topic model’s decomposition of a document as a mixture of topics agrees with human associations of topics with a document". They demonstrated that traditional metrics do not capture whether topics are coherent or not. They argue that their human-oriented experimental evaluations give better results, since the semantic aggregation of topics must be closer to human cognition, not necessarily what mathematics demands.

17.6 Supervised Topic Models

Topic Models, though being a good method, doesn’t suffice! These models are not able to give us predictions with respect to contents that they cluster. For examlpe, if we want to predict movie rates based on the a bunch of reviews. So in Blei and McAuliffe (2008) they assume to have a response variable for each of the contents and the goal is to infer the latent topics predictive of the response variable. However we know that in the previous works, unsupervised topic models like LDA had been used for feature selection, hence for supervised learning, but the clustering in that case work disregard of the output variable. So it might be a good idea to create a joint model for response variable and the LDA clustering. The model consists of the conventional LDA plus a response variable on words and a multivariate normal distribution. The generation from the normal distribution models the correlation between different values. To train the model, a maximum likelihood is found and is simplified using mean-field variational approximation similar to LDA.

17.7 Correspondence LDA

In the paper Blei and Jordan (2003), it aims to model annotated data, model the underlying correlation between samples and annotations by joint distribution of types and conditional distribution of annotated data given types. The paper gives three models: (1) A Gaussian-multinomial mixture model which assumes set of binary latent variables that are used for allocation of clusters. The model is assumed to first generate the binary latent variables and then sample the labels for each data. The second model is based on a variation of LDA Blei et al. (2003), which is called Gaussian-Multinomial LDA. Unlike the previous model, this gives the ability to allocate the label each part of the given sample(like image, or a document) with different labels, with various proportions. In Correspondence Topic Models it combines the flexibility of GM-LDA with GM-Mixture. The model is so much similar to that of GM-LDA but difference is that the image regions and caption words can be considered as conditional on two disjoint sets of factors.

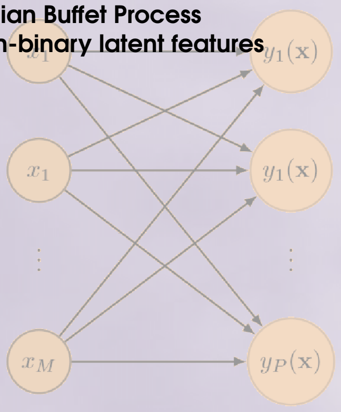
17.8 Inference for Topic Models

There has been various methods suggested for topic modelling, e.g. variational Bayes, Gibbs sampling, ML estimation, MAP estimation, etc. Some of the papers proposing these methods, claim having superior results over the other papers; while Asuncion et al. (2009) shows that all of the methods, in fact give the same structure of answer, having only slight differences on the smoothing which is caused by model hyperparameters, i.e. by careful selection of hyperparameters all of the inference methods almost give a similar answer.

In the rest of the paper they derive the update equations for the models and show that all of the them are equivalent to each other with different hyperparameters.

17.9 Bibliographical notes

In Gibbs sampling I have used Darling (2011). Thanks to Xiaolong Wang's kind helps; I used some parts of his slides.



18 — Bayesian Nonparametrics

18.1 Introduction

[TBW]

18.2 Dirichlet Process

Dichlet Processes(DP) is a crucial element in Bayesian nonparametric modelling; since it is able to model infinite number of components. A DP is infact generalization of Dirichlet Distribution to infinite random variables. To make it clear, if we have set B of disjoint partitions on a set, $\{B_1,...,B_k\}$, we have a DP,

$$G|\alpha,G_0 \sim \mathrm{DP}(\alpha,G_0).$$

in which α is scaling parameter and G_0 is base distribution, such that values of random measures have joint dirhchlet distribution with scaling parameter $\alpha G_0(.)$,

$$(G(B_1),...,G(B_k)) \sim \mathrm{Dir}\left(\alpha G_0(B_1),...,\alpha G_0(B_k)\right).$$

This definition has different interpretations, such as Chinese resturant process, or stick breaking process which are inherently the same. Using the clustering behaviour of DPs one could definite unlimited clusters and use it in infinite mixtor model, similar to conventional mixture models,

$$G|\alpha,G_0 \sim \mathrm{DP}(\alpha,G_0),$$

$$\eta_n|G \sim G,$$

$$X_n|\eta_n \sim p(x_n|\eta_n).$$

Generally these Dirichlet mixture models are trained using MCMC methods. While one could do faster, but approximate learning using mean-field variational approximation. To make everything simple, it is assumed that data is sampled from an exponentional family. In that case, one could assume that G_0, the base distribution of Dirichlet is conjugate prior of the aforementioned exponential family. The approximation is based on the constructive view of DP, i.e. stick breaking process, in which we have an infinite sequence of mixture weights $\pi = \{\pi_i\}_{i=1}^{\infty}$ derived from the following:

$$\beta_i \sim \mathrm{Beta}(1,\alpha)$$

$$\pi_i = \beta_i \prod_{l=1}^{i-1}(1-\beta_l) = \beta_i \left(1 - \sum_{l=1}^{i-1} \pi_l\right)$$

and we denote it by $\pi \sim \text{GEM}(\alpha)$. If we define $G(\theta) = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k}(\theta)$, where $\pi \sim \text{GEM}(\alpha)$ and $\theta_k \sim H$, then one can show that $G \sim \text{DP}(\alpha, H)$.

Assuming the above stick breaking representation, we extend this to a mixture model by adding the following variables:

$$z_i \sim \pi$$

$$\theta_k \sim H(\lambda)$$

$$\mathbf{x}_i \sim F(\theta_{z_i}).$$

So assuming that the parameters in this mixture model are $\mathbf{w} = \{\theta, \pi, \mathbf{z}\}$, we can write the likelihood and simplify it. Let's assume that assume that $\mathbf{w}$ is from an exponential family,

$$p(\mathbf{w}|\mathbf{x}, \xi) = \exp\{\log p(\mathbf{w}, \mathbf{x}|\xi) - p(\mathbf{x}|\xi)\}$$

Using this form, we can derive the necesary relations to minimize the KL lower bound of the real posterior $p(\mathbf{w}|\mathbf{x}, \xi)$ and its approximate value $q(\mathbf{w})$, which can be viewed as maximization of lower bound on marginal likelihood:

$$\text{KL}(q(\mathbf{w})||p(\mathbf{w}|\mathbf{x}, \xi)) = \mathbb{E}_q \log q(\mathbf{W}) - \mathbb{E}_q \log p(\mathbf{W}, \mathbf{x}|\xi) + \log p(\mathbf{x}|\xi)$$

To make the computations tractable, a *truncated* stick breaking process is assumed and the joint distribution of the hyperparameters is

$$q(\mathbf{z}, \theta, \pi) = \prod_{t=1}^{T-1} q_{\gamma_t}(\pi_t) \prod_{t=1}^{T-1} q_{\tau_t(\theta_t)} \prod_{n=1}^{N} q_{\phi_n}(z_n)$$

the first distribution $q_{\gamma_t}(\pi_t)$ is binomial, and the second one is an exponential distribution and the third one is a multinomial distribution. By plugging this distribution in the variational lower bound and optimizing with respect to each of the variables, one could train the model.

18.3 Infinite Hidden Markov Model (iHMM)

In Beal et al. (2002) they've shown an HMM with countably infinite states using Dirichlet processes. They have only three parameters that need to be learnt from data. Conventionally HMM could be trained using Baum-Welch algorithm by taking int account counts of outputs, after specifying the set of possible hidden states. While this parameter optimization would result in over/under fitting the model; thus it might seem a reasonable idea to provide a Bayesian model of HMMs. They use a two-level hierarchical Dirichlet Process model to create infinite state structure.To do inference they do Gibbs sampling which takes quite a long time.

18.4 Indian Buffet Process

This is a review to Griffiths and Ghahramani (2011) which introduces Indian Buffet Process (IBP) which is a tool for non-parametric modelling. In fact IBP is a process which is defined on sparse binary matrices. with finite number of rows and unbounded number of columns.This process is useful as prior in models where there is matrix of data, potentially of size infinity. This could be used to design models with infinite number of hidden features. Or similarly one could assume a binary matrix representation for clustering, in which any data-point could

belong to several clusters.
The mathematical modelling for the elements of the random matrix is as follows:

$$z_{nk} = 1, \text{ iff the} n\text{-th object has} k\text{-th feature.}$$

$$z_{nk} \sim \text{Bernoulli}(\theta_k)$$
$$\theta_k \sim \text{Beta}(\alpha/K, 1)$$

Some properties:

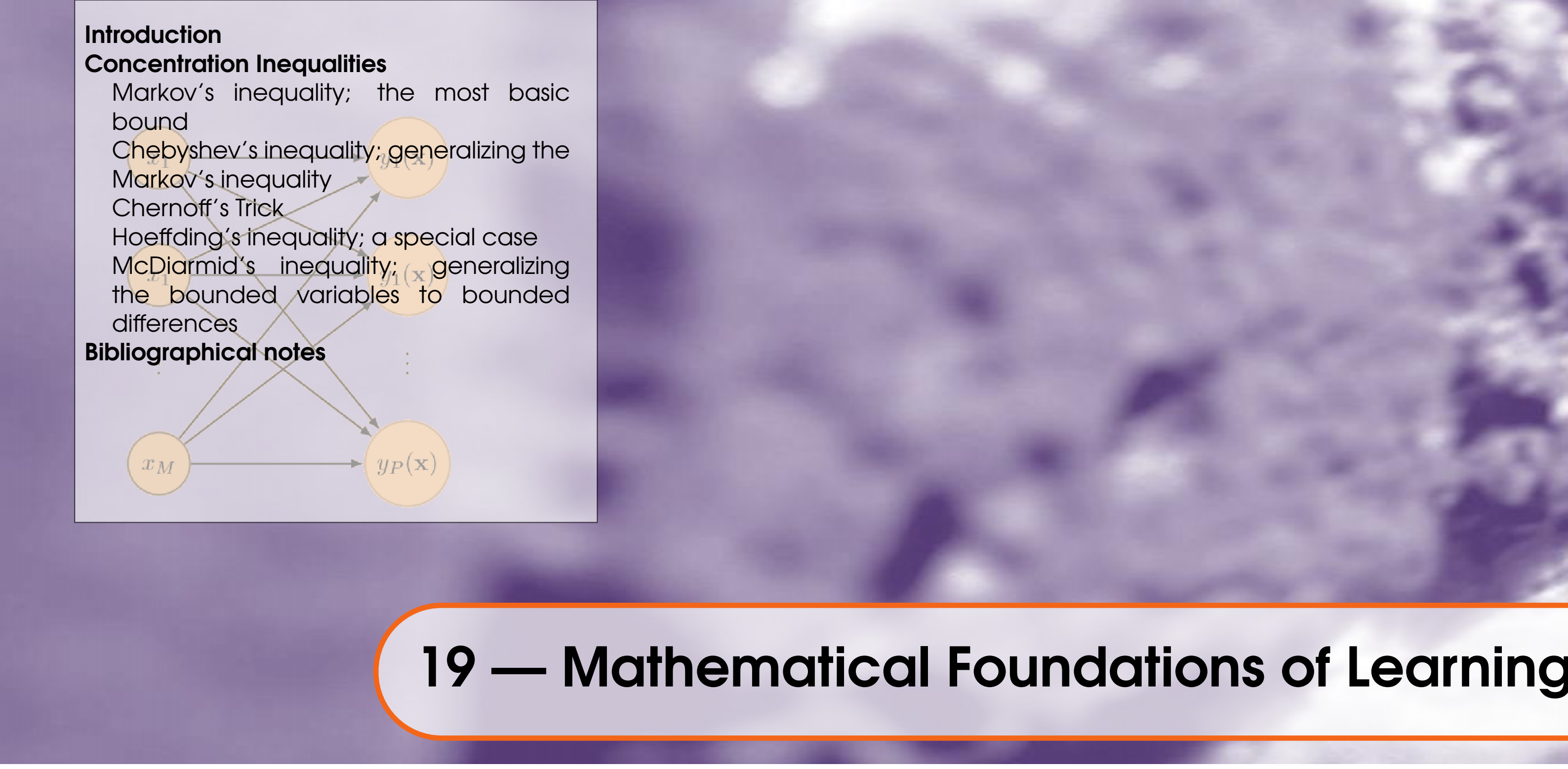
- One could check that $\Pr(z_{nk} = 1|\alpha) = \mathbb{E}(\theta_k) = \frac{\alpha/K}{\alpha/K+1}$. So as K grows, the matrix becomes sparser.
- The expected number of zero elements in $\mathbf{Z} = \frac{N\alpha}{1+\alpha/K} < N\alpha$. Thus even when $K \to \infty$, the number of non-zero elements is still finite.
- It is infinitely exchangeable.
- The expected number of ones in each row is Poisson(α).
- The expected number of ones is αN.
- The number of nonzero columns grow as $O(\alpha \log N)$.

The realization of the above process is called Indian Buffet Process(IBP); the first costumer comes in, takes serving from each dish, stopping after Poisson(α) number of dishes as her plate become overburdened. The nth costumer moves along, sampling the dishes with probability with proportion to their popularity m_k/n, and stops after trying Poisson(α/n) dishes.
Similar to the stick breaking representation of the DP, one could think of stick-breaking representation for IBP.

18.5 non-binary latent features

In most of applications we are dealing with non-binary features. Now can we extend the current model of binary features with infinite dimension, to the case of real features with infinite dimension? We can generate a random matrix **V** of independent random values, of any arbitrary distribution, and find its Hadamard multiplication with the random binary matrix.

Blei et al. (2010) is aiming at solving some problems in hierarchical dependence between topics in the conventional LDA model. In fact, the original topic models models topics as all at the same level, without any directed relationship between one topic and another, like a tree. But it is more realistic to have a level of abstraction of a topic, or how the various topics are related. The goal is nested Chinese restaurant process is to create a hierarchy of topics such that in which more abstract topics are near the root of the hierarchy and more concrete topics are near the leaves. So instead of assuming a line of tables and costumers, they assume having a tree of tables, in which a costumer starts a path from a root and moves long a path, until it reaches a leaf.
They provide a simple sampling procedure for learning and inference in the model. It is important to mention that, being able to move along a path is a limitation for this model, which solved in a subsequent work called “Nested Hierarchical Dirichlet Process".



19 — Mathematical Foundations of Learning

19.1 Introduction

To measure up how the modelling using samples is close to the original (unknown) system, researchers have developed various. One of the ways to quantify this is to use *probabilistic* approach for modelling the *concentration* (closeness) of the model to the original system. Such analysis need using various probabilistic inequalities that could bound the results.

This is an introduction to concentration measures, with a focus on learning algorithms.

19.2 Concentration Inequalities

19.2.1 Markov's inequality; the most basic bound

Theorem 19.1 If X is a non-negative random variable, for any $t > 0$ we have

$$\mathbb{P}(X \geq t) \leq \frac{\mathbb{E}X}{t}$$

Proof.

$$\begin{aligned}
\mathbb{E}X &= \int_{X \in \mathscr{X}} P_X(dx) \\
&= \int_{X \in \mathscr{X}, X<t} X P_X(dx) + \int_{X \in \mathscr{X}, X \geq t} X P_X(dx) \\
&\geq \int_{X \in \mathscr{X}, X \geq t} X P_X(dx) \\
&\geq \int_{X \in \mathscr{X}, X \geq t} t P_X(dx) = t \int_{X \in \mathscr{X}, X \geq t} P_X(dx) = t\mathbb{P}(X \geq t)
\end{aligned}$$

■

19.2.2 Chebyshev's inequality; generalizing the Markov's inequality

Theorem 19.2

$$\mathbb{P}(|X-\mathbb{E}X| \geq t) \leq \frac{\mathrm{Var}X}{t^2}$$

Proof. There is a simple proof using Markov's inequality. Based Markov's inequality we know,

$$\mathbb{P}(X \geq t) = \mathbb{P}(\phi(X) \geq \phi(t)) \leq \frac{\mathbb{E}\phi(X)}{\phi(t)}$$

Now assume $\phi(x) = x^2 (x \geq 0)$, which gives the desired result. In other words, using the Markov's inequality

$$\mathbb{P}(|X-\mathbb{E}X| \geq t) = \mathbb{P}\left(|X-\mathbb{E}X|^2 \geq t^2\right)$$
$$\leq \frac{\mathbb{E}|X-\mathbb{E}X|^2}{t^2} = \frac{\mathbb{E}X^2 - (\mathbb{E}X)^2}{t^2} = \frac{\mathrm{Var}X}{t^2}$$

■

R In general $\phi(x) = x^q (x \geq 0)$, for any positive q we have,

$$\mathbb{P}(|X-\mathbb{E}X| \geq t) \leq \frac{\mathbb{E}|X-\mathbb{E}X|^q}{t^q}$$

Because the goal is to find tighter upper-bounds one can minimize the right-side with respect to q.

■ **Example 19.1** d1 ■

19.2.3 Chernoff's Trick

For any $s > 0$,

$$\mathbb{P}(X > t) = \mathbb{P}(e^{sX} > e^{st}) \leq e^{-st}\mathbb{E}\left[e^{sX}\right]$$

To make the bound as tight as possible,

$$\mathbb{P}(X > t) \leq \inf_{s>0} e^{-st}\mathbb{E}\left[e^{sX}\right]$$

19.2.4 Hoeffding's inequality; a special case

■ **Lemma 19.1** If $X \in \mathscr{X}$ is a random variable with $\mathbb{E}X = 0$, and $\exists a, b \in \mathscr{X}$, s.t. $\mathbb{P}(a \leq X \leq b) = 1$, then for any $s > 0$

$$\mathbb{E}\left[e^{sX}\right] \leq e^{\frac{1}{8}s^2(b-a)^2}$$

Proof. Assume any point $x \in [a,b]$, which can be represented as $x = \beta.b + (1-\beta).a, 0 \leq \beta \leq 1$. Let us assume a function $\phi(x) = e^{sx}$ for any $s > 0$ which is a convex function on $\mathscr{X} = \mathbb{R}$. Then we have,

$$e^{sx} \leq \beta e^{sb} + (1-\beta)e^{sa}$$

By replacing β with $\frac{x-a}{b-a}$, and since $EX = 0$, we have,

$$\mathbb{E}\left[e^{sx}\right] \leq \frac{b}{b-a}e^{sa} - \frac{a}{b-a}e^{sb}$$

■

Theorem 19.3 — Hoeffding's inequality. Let $X_1, \dots, X_n$ be independent random variables, and for any X_i, $\exists a_i, b_i \in \mathscr{X} = \mathbb{R}$, s.t. $\mathbb{P}(a_i \leq X \leq b_i) = 1$. Let $S_n \triangleq \sum_{i=1}^n X_i$ then for any $s > 0$,

$$\mathbb{P}\left(|S_n - \mathbb{E}S_n| \geq t\right) \leq 2\exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right)$$

Proof. We can divide the inequality into two parts,

$$\mathbb{P}\left(S_n - \mathbb{E}S_n \geq t\right) \leq 2\exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right) \tag{19.1}$$

$$\mathbb{P}\left(S_n - \mathbb{E}S_n \leq -t\right) \leq 2\exp\left(-\frac{2t^2}{\sum_{i=1}^n (b_i - a_i)^2}\right) \tag{19.2}$$

To use the Lemma 19.1, we can replace each variable $X_i \to X_i - \mathbb{E}X_i$, so that $\mathbb{E}X_i = 0$; then we have, $\mathbb{E}\left[e^{sX_i}\right] \leq e^{s^2(b_i - a_i)^2/8}$. Using Chernoff's trick,

$$\mathbb{P}\left(S_n \geq t\right) = \mathbb{P}\left(e^{S_n} \geq e^t\right) \leq e^{-st}\mathbb{E}\left[e^{sS_n}\right]$$

And since X_i's are independent,

$$\mathbb{E}\left[e^{sS_n}\right] = \mathbb{E}\left[e^{s(X_1 + \ldots + X_n)}\right] = \mathbb{E}\left[\prod_{i=1}^n e^{sX_i}\right] = \prod_{i=1}^n \mathbb{E}\left[e^{sX_i}\right]$$

$$\Rightarrow \mathbb{P}\left(S_n \geq t\right) \leq e^{-st}\prod_{i=1}^n e^{s^2(b_i - a_i)^2/8} = \exp\left\{-st + \frac{s^2}{8}\sum_{i=1}^n (b_i - a_i)^2\right\}$$

To minimize the right hand-side with respect to s we can choose it to be $s = \frac{4t}{\sum_{i=1}^n (b_i - a_i)^2}$. This proves the Equation 19.1. The proof for the Equation 19.2 is similar. ■

19.2.5 McDiarmid's inequality; generalizing the bounded variables to bounded differences

Definition 19.1 — A function with bounded differences. A function is said to have bounded differences, if changing only one variable, and keep everything the save, the absolute value of the difference is always bounded. In other words, if we have a function $f : \mathscr{X}^n \to \mathbb{R}$, then,

$$\exists c_i \in \mathbb{R}, \text{ s.t. } \forall x_1, \ldots, x_n, x_i' \in \mathbb{X}, \sup\left|f(x_1, \ldots, x_n) - f(x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_n)\right| \leq c_i$$

Theorem 19.4 — McDiarmid's inequality. Let $X = (X_1, \ldots, X_n) \in \mathscr{X}^n$, and $f : \mathscr{X}^n \to \mathbb{R}$ has bounded differences. Then for any $t > 0$,

$$\mathbb{P}\left(|f(X) - \mathbb{E}f(X)| \geq t\right) \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n c_i^2}\right)$$

Proof. The final result can be broken into two parts,

$$\mathbb{P}\left(f(X) - \mathbb{E}f(X) \geq t\right) \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n c_i^2}\right),$$

and

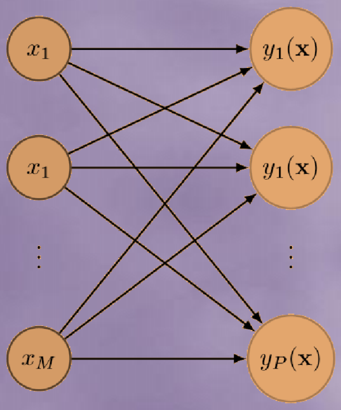
$$\mathbb{P}\left(\mathbb{E}f(X) - f(X) \geq t\right) \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n c_i^2}\right).$$

■

[More details: TBW]

19.3 Bibliographical notes

In preparation of this I have used Raginsky (2011).



Bibliography

Yasemin Altun and Alex Smola. Unifying divergence minimization and statistical inference via convex duality. In *Learning theory*, pages 139–153. Springer, 2006.

Arthur Asuncion, Max Welling, Padhraic Smyth, and Yee Whye Teh. On smoothing and inference for topic models. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 27–34. AUAI Press, 2009.

Matthew J Beal, Zoubin Ghahramani, and Carl Edward Rasmussen. The infinite hidden markov model. *Advances in neural information processing systems*, 14:577–584, 2002.

C.M. Bishop. *Pattern recognition and machine learning*, volume 4. springer New York, 2006.

D. Blei and J. Lafferty. Correlated topic models. *Advances in neural information processing systems*, 18:147, 2006.

David Blei and Jon McAuliffe. Supervised topic models. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 121–128. MIT Press, Cambridge, MA, 2008.

David M Blei and Michael I Jordan. Modeling annotated data. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 127–134. ACM, 2003.

David M Blei, Thomas L Griffiths, and Michael I Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *Journal of the ACM (JACM)*, 57 (2):7, 2010.

D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

Jonathan Boyd-Graber, Jordan Chang, Sean Gerrish, Chong Wang, and David Blei. Reading tea leaves: How humans interpret topic models. In *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems*, 2009.

Volkan Cevher. *Variational Bayes Approximation*. Rice University, 2008. Lecture notes of Graphical Models course.

William M Darling. A theoretical and practical implementation tutorial on topic modeling and gibbs sampling. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 642–647, 2011.

Brendan J Frey. Extending factor graphs so as to unify directed and undirected graphical models. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 257–264. Morgan Kaufmann Publishers Inc., 2002.

Kuzman Ganchev, Joao Graça, Jennifer Gillenwater, and Ben Taskar. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research*, 99:2001–2049, 2010.

W.R. Gilks and P. Wild. Adaptive rejection sampling for gibbs sampling. *Applied Statistics*, pages 337–348, 1992.

W.R. Gilks, NG Best, and KKC Tan. Adaptive rejection metropolis sampling within gibbs sampling. *Applied Statistics*, pages 455–472, 1995.

Thomas L Griffiths and Zoubin Ghahramani. The indian buffet process: An introduction and review. *Journal of Machine Learning Research*, 12:1185–1224, 2011.

W.K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.

JONATHAN Huang and TOMASZ Malisiewicz. Fitting a hierarchical logistic normal distribution, 2009.

M. Jeruchim. Techniques for estimating the bit error rate in the simulation of digital communication systems. *Selected Areas in Communications, IEEE Journal on*, 2(1):153–170, 1984.

D. Koller, U. Lerner, and D. Angelov. A general algorithm for approximate inference and its application to hybrid bayes nets. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 324–333. Morgan Kaufmann Publishers Inc., 1999.

Frank R Kschischang, Brendan J Frey, and H-A Loeliger. Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on*, 47(2):498–519, 2001.

S. Kullback and R.A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

Malte Kuss and Carl Edward Rasmussen. Assessing approximate inference for binary gaussian process classification. *The Journal of Machine Learning Research*, 6:1679–1704, 2005.

S.L. Lauritzen. Propagation of probabilities, means, and variances in mixed graphical association models. *Journal of the American Statistical Association*, pages 1098–1108, 1992.

P.S. Maybeck. *Stochastic models, estimation and control*, volume 1. Academic Pr, 1979.

N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21:1087, 1953.

Thomas Minka. Power ep. *Dep. Statistics, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep*, 2004.

Thomas P Minka. The ep energy function and minimization schemes. *See www. stat. cmu. edu/˜ minka/papers/learning. html*, 2001a.

Thomas P Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 362–369. Morgan Kaufmann Publishers Inc., 2001b.

Tom Minka et al. Divergence measures and message passing. *Microsoft Research Cambridge, Tech. Rep. MSR-TR-2005-173*, 2005.

T.P. Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001c.

Manfred Opper, Ulrich Paquet, and Ole Winther. Improving on expectation propagation. *Advances in Neural Information Processing Systems (NIPS)*, 2008:8–13, 2008.

Jason L Pacheco and Erik B Sudderth. Improved variational inference for tracking in clutter. In *Statistical Signal Processing Workshop (SSP), 2012 IEEE*, pages 852–855. IEEE, 2012.

Yuan Qi and Yandong Guo. Message passing with l1 penalized kl minimization. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 262–270. JMLR Workshop and Conference Proceedings, May 2013. URL `http://jmlr.org/proceedings/papers/v28/qi13.pdf`.

A.E. Raftery and S. Lewis. How many iterations in the gibbs sampler. *Bayesian statistics*, 4(2): 763–773, 1992.

Maxim Raginsky. Lecture notes: Ece 299: Statistical learning theory. *Tutorial*, 2011. URL `http://maxim.ece.illinois.edu/teaching/spring11/schedule.html`.

year = 2012 url = http://l2r.cs.uiuc.edu/ danr/Teaching/CS546-13/ Ran Roth, title = CS546 at UIUC: Machine Learning for NLP, Homework.

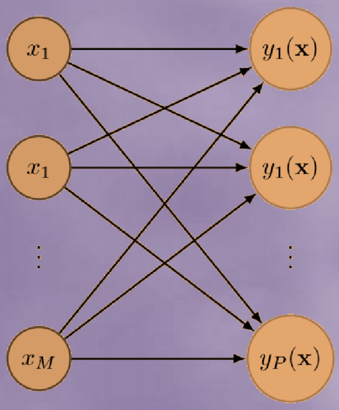
Matthias W Seeger and Hannes Nickisch. Fast convergent algorithms for expectation propagation approximate bayesian inference. *arXiv preprint arXiv:1012.3584*, 2010.

A.F.M. Smith and G.O. Roberts. Bayesian computation via the gibbs sampler and related markov chain monte carlo methods. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 3–23, 1993.

Charles Sutton. Expectation propagation in factor graphs:a tutorial. 2005.

Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.

Wim Wiegerinck, Tom Heskes, et al. Fractional belief propagation. *Advances in Neural Information Processing Systems*, pages 455–462, 2003.



Index