Engineering Tripos Part IIA THIRD YEAR

Paper 3F8: Inference

Solutions to Example Sheet 2: Bayesian Linear Regression, Classification, Dimensionality Reduction and Clustering

Bayesian linear regression

1. Bayesian Linear Regression

   A single data point $\{x, y\}$ is fit using Bayesian linear regression. The output y is assumed to be generated from the input x according to a linear relationship that is corrupted by Gaussian noise $y = mx + c + \epsilon$. The noise ϵ is mean 0 and variance 1 so $p(y|m, c, x) = \mathcal{N}(y; mx + c, 1)$. Gaussian priors are placed on the slope m and intercept c with zero mean and unit variance, that is $p(m) = \mathcal{N}(m; 0, 1)$ and $p(c) = \mathcal{N}(c; 0, 1)$.

(a) Compute the posterior probability of the slope and intercept given the data point, that is $p(m, c|x, y)$.

a) $p(m,c \mid y,x) = \dfrac{p(y \mid m,c,x)\,p(m)p(c)}{p(y \mid x)}$

$\propto p(y \mid m,c,x)\,p(m)p(c)$ ← product of Gaussian functions in m & c

$= \mathcal{N}\left(\begin{bmatrix} m \\ c \end{bmatrix}; \underline{\mu}^{post}, \underline{\underline{\Sigma}}^{post} \right)$ ← posterior must be Gaussian: just need to compute mean & covariance

tactic

① Substitute in for these densities and ② identify mean & covariance by comparing coefficients

① $p(m,c \mid y,x,\sigma_y^2) \propto \dfrac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(y-mx-c)^2} \dfrac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}m^2} \dfrac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}c^2}$

$\propto e^{-\frac{1}{2}y^2 - \frac{1}{2}m^2(x^2+1) - \frac{1}{2}c^2 \times 2 - mcx + y(mx+c)}$

② $\propto e^{-\frac{1}{2}\left(\begin{bmatrix} m \\ c \end{bmatrix} - \underline{\mu}^{post} \right)^T \left(\underline{\underline{\Sigma}}^{post} \right)^{-1} \left(\begin{bmatrix} m \\ c \end{bmatrix} - \underline{\mu}^{post} \right)}$

$\propto e^{-\frac{1}{2} \begin{bmatrix} m \\ c \end{bmatrix}^T \left(\underline{\underline{\Sigma}}^{post} \right)^{-1} \begin{bmatrix} m \\ c \end{bmatrix} + \begin{bmatrix} m \\ c \end{bmatrix}^T \left(\underline{\underline{\Sigma}}^{post} \right)^{-1} \underline{\mu}^{post}}$

$\begin{bmatrix} m \\ c \end{bmatrix}^T \begin{bmatrix} yx \\ y \end{bmatrix}$ inverse covariance times mean

inverse Covariance

$\Rightarrow \underline{\underline{\Sigma}}^{post} = \begin{bmatrix} x^2+1 & x \\ x & 2 \end{bmatrix}^{-1} = \dfrac{1}{(x^2+1)(2) - x^2} \begin{pmatrix} 2 & -x \\ -x & x^2+1 \end{pmatrix}$

$= \dfrac{1}{x^2+2} \begin{pmatrix} 2 & -x \\ -x & x^2+1 \end{pmatrix}$

& since $\left(\underline{\underline{\Sigma}}^{post} \right)^{-1} \underline{\mu}^{post} = \begin{bmatrix} yx \\ y \end{bmatrix}$

$\underline{\mu}^{post} = \dfrac{1}{x^2+2} \begin{pmatrix} 2 & -x \\ -x & x^2+1 \end{pmatrix} \begin{bmatrix} yx \\ y \end{bmatrix}$ $\underline{\underline{\Sigma}}^{post}$

$= \dfrac{1}{x^2+2} \begin{bmatrix} 2yx - yx \\ -yx^2 + (x^2+1)y \end{bmatrix} = \dfrac{1}{x^2+2} \begin{bmatrix} yx \\ y \end{bmatrix}$

(b) Show that the posterior derived in part a is consistent with the expressions for Bayesian linear regression given in lectures.

b) Check these expressions match with those in lectures: $\underline{y} = \tilde{\underline{\underline{X}}}\,\underline{w} + \underline{\epsilon}$ $\quad \underline{\epsilon} \sim N(\underline{0}, \sigma^2 \underline{\underline{I}})$

$p(\underline{w} \mid \text{Data}, \sigma^2, \lambda) = N(\underline{w};\ \underline{\mu}_{w|D},\ \underline{\underline{\Sigma}}_{w|D})$ $\quad \underline{w} \sim N(\underline{0}, \lambda^{-1}\underline{\underline{I}})$

where $\underline{\underline{\Sigma}}_{w|D} = \left(\underline{\underline{I}}\lambda + \frac{1}{\sigma^2}\tilde{\underline{\underline{X}}}^T\tilde{\underline{\underline{X}}}\right)^{-1}$ $\quad \underline{\mu}_{w|D} = \underline{\underline{\Sigma}}_{w|D}\,\frac{1}{\sigma^2}\tilde{\underline{\underline{X}}}^T\underline{y}$

in our case $\sigma^2 = \lambda = 1$ (prior & observation noise are unit variance)

$\tilde{\underline{\underline{X}}} = [x, 1]$

$\underline{w} = \begin{bmatrix} m \\ c \end{bmatrix}$

$\Rightarrow \quad \underline{\underline{\Sigma}}_{w|D} = \left(\underline{\underline{I}} + \begin{bmatrix} x \\ 1 \end{bmatrix}[x, 1]\right)^{-1} = \begin{pmatrix} x^2+1 & x \\ x & 2 \end{pmatrix}^{-1}$ ✓

$\underline{\mu}_{w|D} = \begin{pmatrix} x^2+1 & x \\ x & 2 \end{pmatrix}^{-1} \begin{bmatrix} x \\ 1 \end{bmatrix} y$ ✓

(c) Compute the posterior in the following three cases and provide explanations for why the posteriors take the form that they do.

i. $x = 0$ and $y = 0$

ii. $x = 1$ and $y = 0$

iii. $x = 100$ and $y = 100$

c) Plug the following settings $x=0, y=0$ $\quad x=1, y=1$ $\quad x=100, y=100$

into:

$$\Sigma^{post} = \frac{1}{x^2+2}\begin{bmatrix} 2 & -x \\ -x & x^2+1 \end{bmatrix}$$

$$\mu^{post} = \frac{1}{x^2+2}\begin{bmatrix} yx \\ 2y \end{bmatrix}$$

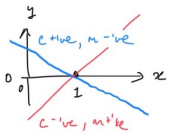
i) $x=0$ $y=0$ $\quad \Sigma^{post} = \begin{bmatrix} 1 & 0 \\ 0 & 1/2 \end{bmatrix}$ $\quad \mu^{post} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

ii) $x=1$ $y=0$ $\quad \Sigma^{post} = \begin{bmatrix} 2/3 & -1/3 \\ -1/3 & 2/3 \end{bmatrix}$ $\quad \mu^{post} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

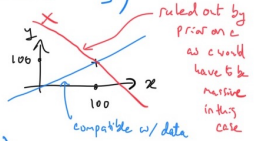
iii) $x=100$ $y=100$ $\quad \Sigma^{post} \approx \begin{bmatrix} 2/100^2 & -1/100 \\ -1/100 & 1 \end{bmatrix}$ $\quad \mu^{post} \approx \begin{bmatrix} 1 \\ 1/100 \end{bmatrix}$

i) data doesn't tell us anything about m: $y = mx + c$ & $x = 0$
This is why posterior mean & variance of m stays @ prior (mean 0 variance 1)
data tells us that c is likely to be ≈ 0, but the observation noise is quite large so only weak evidence for this. This why posterior mean is 0 and posterior variance is $1/2$ (prior variance was 1).

ii) The data can be explained by either positive c & negative m or negative c & positive m:
Hence posterior covariance between m & c is negative $(-1/3)$
Dataset provides weak evidence that m & c are both small in magnitude & due to symmetry both still have posterior mean 0 (uncertainty has reduced from variance 1 to variance $2/3$)



iii) The data $y=100$ $x=100$ rules out negative m:
We are therefore quite certain that m is close to 1
(posterior mean very close to this value & variance $\approx 2/100^2$)
The data give almost no information about c though with posterior mean & variance close to the prior.



You might like to compute the predictive mean for these three cases i.e. the mean of $p(y^*|x^*, x, y)$ where x^* is the location of a new test input and y^* is the corresponding output.

Predictive mean: use $y^* = m x^* + c + \varepsilon^*$ now average wrt $p(m, c, \varepsilon^* | x, y)$:

$$\mathbb{E}_{p(m,c,\varepsilon^*|x,y)}\left[m x^* + c + \varepsilon^* \right] = \mu_m^{post} x^* + \mu_c^{post} + 0$$

(independent from x & y)

i) predictive mean $= 0$ (horizontal line)

ii) predictive mean $= 0$ (horizontal line)

iii) predictive mean $= \frac{1}{100^2 + 2}\left(100^2 x^* + 100\right)$ (line with gradient v. close to 1 and a small positive intercept)

Classification

2. Probit Classification

Consider classification as described in lectures, but with the following model

$$y^{(n)} = H(\mathbf{w}^\mathsf{T}\mathbf{x}^{(n)} + \epsilon_n)$$

where ϵ_n is Gaussian with mean 0 and variance σ^2 and $H(\cdot)$ is the Heaviside step function.

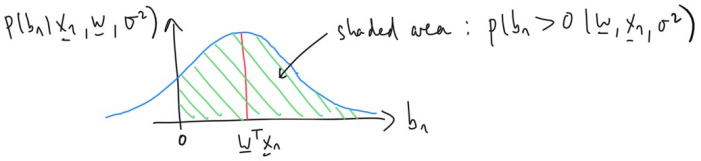
(a) Compute the probability $P(y^{(n)} = 1|\mathbf{x}_n, \mathbf{w}, \sigma^2)$ in terms of the Gaussian cumulative distribution. Sketch $P(y^{(n)} = 1|\mathbf{x}_n, \mathbf{w}, \sigma^2)$ as a function of the inputs $\mathbf{x}_n$ in the case where they are one dimensional.

(b) What happens as the noise variance tends to infinity $\sigma^2 \to \infty$?

a) $y_n = H(\underline{w}^T \underline{x}_n + \epsilon_n) \qquad \epsilon_n \sim N(0, \sigma^2)$

let $b_n = \underline{w}^T \underline{x}_n + \epsilon_n \qquad b_n \sim N(\underline{w}^T \underline{x}_n, \sigma^2)$

$\therefore \quad p(y_n = 1 \mid \underline{w}, \underline{x}_n, \sigma^2) = p(b_n > 0 \mid \underline{w}, \underline{x}_n, \sigma^2)$



$\therefore \quad p(y_n = 1 \mid \underline{w}, \underline{x}_n, \sigma^2) = \int_0^\infty N(b_n ; \underline{w}^T \underline{x}_n, \sigma^2)\, db_n$

let $v_n = -\left(\frac{b_n - \underline{w}^T \underline{x}_n}{\sigma}\right)$ & transform to integral over this quantity (v_n will be distributed according to a standard normal)

$$p(y_n = 1 \mid \underline{w}, \underline{x}_n, \sigma^2) = \int_{-\infty}^{\frac{\underline{w}^T \underline{x}_n}{\sigma}} N(v_n ; 0, 1)\, dv_n$$

$$p(y_n = 1 \mid \underline{w}, \underline{x}_n, \sigma^2) = \text{CDF}\left(\frac{\underline{w}^T \underline{x}_n}{\sigma}\right)$$

b) $\sigma^2 \to \infty \quad \Rightarrow p(y_n = 1 \mid \underline{x}_n, \underline{w}, \sigma^2) = 1/2$

The noise swamps $\underline{w}^T \underline{x}_n$ resulting in an output which is a coin toss

3. Multi-class Classification

Consider a multi-class classification problem with K classes. The training labels are represented by K dimensional vectors $\mathbf{t}_n$ which has a single element set to 1, indicating the class membership, and all other values are set to 0. The inputs are multi-dimensional vectors $\mathbf{x}_n$. The goal is to use a training set of input vectors and output labels $\{\mathbf{x}_n, \mathbf{t}_n\}_{n=1}^N$ to enable prediction at unseen input locations.

A friend suggests using a a soft-max function for this purpose which is parameterised by weights $\mathbf{W} = \{\mathbf{w}_k\}_{k=1}^K$. The output of the function is a vector, $\mathbf{y}$, with elements given by

$$y_i(\mathbf{x}; \mathbf{W}) = \frac{\exp(\mathbf{w}_i^\mathsf{T}\mathbf{x})}{\sum_{k=1}^K \exp(\mathbf{w}_k^\mathsf{T}\mathbf{x})}.$$

(a) What happens to the softmax function as the magnitude of the weights tends to infinity?

(b) Interpretting the output of the softmax as $y_i = p(\mathrm{t}_i = 1|\mathbf{W}, \mathbf{x})$ write down a cost-function for training this network based on the log-probability of the training data given the weights $\mathbf{W}$ and inputs $\{\mathbf{x}_n\}_{n=1}^N$.

(c) What is the relationship between this network and logistic regression?

$$y_i(\underline{x};\underline{\underline{w}}) = \frac{e^{\underline{w}_i^T \underline{x}}}{\sum_{k=1}^{K} e^{\underline{w}_k^T \underline{x}}}$$

let $\hat{\underline{w}}_{max} = \underset{\hat{\underline{w}}_i \in \hat{\underline{w}}_1 \dots \hat{\underline{w}}_K}{\text{argmax}}\ \hat{\underline{w}}_i^T \underline{x}$

a) let $\underline{w}_i = \beta \hat{\underline{w}}_i$ ← unit vector in direction of $\underline{w}_i$
↑ magnitude

$$y_i(\underline{x};\underline{\underline{w}}) = \frac{e^{\beta \hat{\underline{w}}_i^T \underline{x}}}{\sum_{k=1}^{K} e^{\beta \hat{\underline{w}}_k^T \underline{x}}} = \frac{e^{\beta(\hat{\underline{w}}_i - \hat{\underline{w}}_{max})^T \underline{x}}}{\sum_{k=1}^{K} e^{\beta(\hat{\underline{w}}_k - \hat{\underline{w}}_{max})^T \underline{x}}}$$

$(\hat{\underline{w}}_i - \hat{\underline{w}}_{max})^T \underline{x}$ are a set of K scalars, one of which is zero & the others are negative

$\therefore$ as $\beta \to \infty$ $\quad e^{\beta(\hat{\underline{w}}_i - \hat{\underline{w}}_{max})} \to \begin{cases} 0 & \text{if } \hat{\underline{w}}_i \neq \underline{w}_{max} \\ 1 & \text{if } \hat{\underline{w}}_i = \underline{w}_{max} \end{cases}$

$$\Rightarrow \quad y_i(\underline{x};\underline{\underline{w}}) \to \begin{cases} 0 & \text{if } i \neq i_{max} \\ 1 & \text{if } i = i_{max} \end{cases}$$

hence the general name "softmax"

This is a zero/one encoding of the argmax function:

ie as $\beta \to \infty$ $\quad y_i(\underline{x};\underline{\underline{w}}) \to \mathbb{I}\left(\underset{k}{\text{argmax}}\ \hat{\underline{w}}_k^T \underline{x} = i \right)$

↑ indicator function that takes the value 1 if the argument is true & is otherwise 0.

NB
Many people will be able to intuit this result without going through all of this.

b) $y_i = p(t_i = 1 \mid \underline{\underline{W}}, \underline{x})$

$$p\left(\{\underline{t}^{(n)}\}_{n=1}^{N} \mid \underline{\underline{W}}, \{\underline{x}^{(n)}\}_{n=1}^{N}\right) = \prod_{n=1}^{N} p(\underline{t}^{(n)} \mid \underline{\underline{W}}, \underline{x}^{(n)})$$

$$= \prod_{n=1}^{N} \prod_{k=1}^{K} y_k(\underline{x}^{(n)}, \underline{\underline{W}})^{t_k^{(n)}}$$

$$= e^{\sum_n \sum_k t_k^{(n)} \log y_k(\underline{x}^{(n)}, \underline{\underline{W}})}$$

$\Rightarrow$ cost function could be

$$\underset{\underline{\underline{W}}}{\arg\max} \sum_{n=1}^{N} \sum_{k=1}^{K} t_k^{(n)} \log y_k(\underline{x}^{(n)}, \underline{\underline{W}})$$

c) Consider $K = 2$ (two classes)

$$y_1(\underline{x}, \underline{\underline{W}}) = \frac{e^{\underline{w}_1^T \underline{x}}}{e^{\underline{w}_1^T \underline{x}} + e^{\underline{w}_2^T \underline{x}}} = \frac{1}{1 + e^{(\underline{w}_2 - \underline{w}_1)^T \underline{x}}}$$

$$= \frac{1}{1 + e^{-\underline{v}^T \underline{x}}}$$

$\Rightarrow$ equivalent to logistic classification when there are 2 classes

Dimensionality Reduction

4. Principal Component Analysis†

A dataset comprises of N data points $\{x_n\}_{n=1}^N$. The data are zero-mean, $\mu = \frac{1}{N}\sum_{n=1}^N x_n = 0$, and the data-covariance is given by, $\Sigma = \frac{1}{N}\sum_{n=1}^N x_n x_n^\top$.

(a) Describe the purpose of the principal component analysis algorithm and outline one procedure for computing the first principal component of a dataset.

(b) Consider a data-covariance $\Sigma = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$. Show that the two eigenvectors for this matrix are $\frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and $\frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix}$ and state the corresponding eigenvalues. What is the first principal component when $-1 < \rho < 0$?

a) PCA carries out dimensionality reduction

Find eigenvector of covariance matrix Σ with the highest eigenvalue

ie find $\underline{\underline{\Sigma}}\,\underline{e}_\mu = \lambda_\mu \underline{e}_\mu$ principal component $= \underline{e}_{\mu_{max}}$ $\mu_{max} = \underset{\mu}{\operatorname{argmax}} \lambda_\mu$

b) $\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1+\rho \\ \rho+1 \end{bmatrix} = (1+\rho)\frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix}$

$\lambda = 1+\rho$

$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1-\rho \\ \rho-1 \end{bmatrix} = (1-\rho)\frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix}$

$\lambda = 1-\rho$

if $-1 < \rho < 0$ then $\lambda_{max} = 1-\rho \Rightarrow \underline{e}_{max} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix}$

5. Principal Component Analysis and Auto-Encoders**

A data-scientist would like to summarise high dimensional data points $\mathbf{y}_n$ in terms of a single scalar variable x_n. They use an encoding weight $\mathbf{w}$ to produce the summary, $x_n = \mathbf{w}^\top \mathbf{y}_n$, and a decoding weight $\mathbf{r}$ to reconstruct the data point from the summary, $\hat{\mathbf{y}}_n = \mathbf{r}x_n$. This architecture is called an auto-encoder. The data-scientist would like to learn the encoding and decoding weights by optimising the squared error of the reconstruction,

$$\mathcal{C} = \sum_n ||\mathbf{y}_n - \hat{\mathbf{y}}_n||^2.$$

(a) Minimise the cost $\mathcal{C}$ with respect to the decoding weights $\mathbf{r}$, returning an expression for them in terms of x_n and $\mathbf{y}_n$.

(b) Substitute your expression for the optimised decoding weights $\mathbf{r}$ into $\mathcal{C}$ to obtain the cost purely in terms of the encoding weights $\mathbf{w}$.

(c) Now consider minimising the cost derived in part (b) with respect to the encoding weights. What is the solution? Is it unique?

It may be useful to know that the solution to the optimisation problem $\mathbf{z}^* = \arg\max_{\mathbf{z}} \frac{\mathbf{z}^\top H \mathbf{z}}{\mathbf{z}^\top \mathbf{z}}$ is the largest eigenvector of matrix H (arbitrarily scaled), $\mathbf{z}^* \propto \mathbf{e}_1$.

a)

$$\mathcal{C}(\underline{w}, \underline{r}) = \sum_n \sum_d \left(y_{dn} - r_d x_n \right)^2$$

$$\frac{d\mathcal{C}(\underline{w}, \underline{r})}{dr_d} = \sum_n 2\left(y_{dn} - r_d x_n \right) x_n$$

$$\Rightarrow \quad r_d = \frac{\sum_n y_{dn} x_n}{\sum_n x_n^2}$$

(just like linear regression)

$$C(\underline{w}, \underline{r}(\underline{w})) = \sum_n \sum_d \left(y_{nd} - \underbrace{\sum_{n'} y_{n'd} x_{n'}}_{\sum_a x_a^2} x_n \right)^2$$

this more direct using vector notation, but here I use index notation

$$= \sum_{nd} y_{nd}^2 - 2 \sum_n \sum_{n'} \sum_d y_{nd} y_{n'd} \underbrace{x_n x_{n'}}_{\sum_a x_a^2} + \sum_n \sum_{n'} \sum_{n''} y_{n'd} y_{n''d} \underbrace{x_{n'} x_{n''} x_n^2}_{\left(\sum_a x_a^2\right)^2}$$

↓ does not depend on $\underline{w}$

$$= \sum_{nd} y_{nd}^2 - \sum_{nn'd} y_{nd} y_{n'd} \underbrace{x_n x_{n'}}_{\sum_q x_q^2}$$

$\sum_n \underline{y}_n \underline{y}_n^T$ → $\underline{w}^T \underline{\Sigma}_y \underline{\Sigma}_y \underline{w}$ → $\underline{w}^T \underline{\Sigma}_y \underline{w}$

$$\Rightarrow \underset{\underline{w}}{\arg\max} \sum_{nn'd} y_{nd} y_{n'd} \underbrace{x_n x_{n'}}_{\sum_q x_q^2} = \frac{\sum_{nn'} \underline{y}_n^T \underline{y}_{n'} \underline{w}^T \underline{y}_n \underline{w}^T \underline{y}_{n'}}{\sum_a \underline{w}^T \underline{y}_a \underline{y}_a \underline{w}} = \frac{\underline{w}^T \underline{\Sigma}_y \underline{\Sigma}_y \underline{w}}{\underline{w}^T \underline{\Sigma}_y \underline{w}}$$

↖ $\underline{w}^T \underline{\Sigma}_y \underline{w}$

c) Now use $\underline{v} = \underline{\Sigma}_y^{1/2} \underline{w}$ & solve for $\underline{v}$ to find $\underline{w}$ via $\underline{w} = (\underline{\Sigma}_y^{1/2})^{-1} \underline{v}$

$$\underset{\underline{v}}{\arg\max} \frac{\underline{v}^T \underline{\Sigma}_y \underline{v}}{\underline{v}^T \underline{v}}$$

$\Rightarrow \underline{v} \propto$ largest eigenvector of $\underline{\Sigma}_y = \underline{e}_1$

$\Rightarrow \underline{w} \propto$ largest eigenvector of $\underline{\Sigma}_y$ too $= \underline{e}_1$

as $\underline{\Sigma}_y = \underline{E}^T \underline{\Lambda} \underline{E}$ (matrix of eigenvectors; matrix w/ eigenvalues along diagonal)

$\underline{\Sigma}_y^{1/2} = \underline{E}^T \underline{\Lambda}^{1/2} \underline{E}$ share eigenvectors

$\underline{\Sigma}_y^{-1/2} = \underline{E}^T \underline{\Lambda}^{-1/2} \underline{E}$ share eigenvectors

$\Rightarrow \underline{w} \propto \underline{\Sigma}_y^{-1/2} \underline{e}_1 \propto \underline{e}_1$

Not unique as free to scale $\underline{w}$ arbitrarily & rescale $\underline{r}$ accordingly to compensate

Clustering

6. K-means clustering*

   Consider the K-means algorithm that seeks to minimise the cost function

$$C = \sum_{n=1}^{N} \sum_{k=1}^{K} s_{nk} ||x_n - m_k||^2$$

   where m_k is the mean (centre) of cluster k, x_n is the data point n, $s_{nk} = 1$ indicates that the nth data point has been assigned to cluster k, and $s_{nk} = 0$ indicates that it has not been assigned to that cluster. There are N data points and K clusters.

   (a) Given all the assignments $\{s_{nk}\}$ determine the value of m_k that minimises the cost C and give an interpretation in terms of the K-means algorithm.

   (b) You would like to automatically learn the number of clusters K from data. One possibility is to minimise the cost C as a function of K. Explain whether this is a good idea or not, and what the solution to this minimisation is.

   (c) In many real-world applications, data points arrive sequentially and one wants to cluster them as they come in. Devise a sequential variant of the k-means algorithm which takes in one data point at a time and updates the means $\{m_1, ..., m_K\}$ sequentially without revisiting previous data points. Describe your sequential algorithm.

$$C(\{s\}, \{m\}) = \sum_{n=1}^{N} \sum_{k=1}^{K} s_{nk} \| x_n - m_k \|^2 = \sum_{n} \sum_{k} \sum_{d} s_{nk} (x_{nd} - m_{kd})^2$$

a) $$\frac{dC(\{s\}, \{m\})}{dm_{le}} = -2 \sum_{n=1}^{N} s_{nl} (x_{ne} - m_{le}) = 0$$

$$\Rightarrow \sum_{n=1}^{N} s_{nl} x_{ne} = m_{le} \sum_{n=1}^{N} s_{nl}$$

$$\Rightarrow m_{le} = \frac{\sum_{n=1}^{N} s_{nl} x_{ne}}{\sum_{n=1}^{N} s_{nl}}$$

Or in vector form $$\underline{m}_k = \frac{\sum_{n=1}^{N} s_{nk} \underline{x}_n}{\sum_{n=1}^{N} s_{nk}}$$

Sum of all of the data values assigned to k^{th} cluster

number of data points assigned to k^{th} cluster

b) Bad idea : if $K = N$ & $\underline{m}_n = \underline{x}_n$ cost is zero which is lowest possible

i.e. each datapoint is a "cluster"

c) Several possible options for this one but a logical soln would use:

since $\underline{\mu}_k = \dfrac{\sum_{n=1}^{N} \underline{x}_n s_{nk}}{\sum_{n=1}^{N} s_{nk}} = \dfrac{\sum_{n=1}^{N-1} \underline{x}_n s_{nk} + \underline{x}_N s_{Nk}}{\sum_{n=1}^{N-1} s_{nk} + s_{Nk}}$

where $\sum_{n=1}^{N-1} \underline{x}_n s_{nk} = \underline{\beta}_k^{(N-1)}$ and $\sum_{n=1}^{N-1} s_{nk} = \alpha_k^{(N-1)}$

New datum arrives

- assign to nearest cluster and set assignment vector $\{s_{Nk}\}_{k=1}^{K}$
- update statistics $\underline{\beta}_k^{(N)} = \underline{\beta}_k^{(N-1)} + \underline{x}_N s_{Nk}$ [running sum of data vectors associated w/ k^{th} cluster]

  $\alpha_k^{(N)} = \alpha_k^{(N-1)} + s_{Nk}$ [running count of # of data points associated w/ k^{th} cluster]
- recompute means $\underline{\mu}_k = \dfrac{\underline{\beta}_k^{(N)}}{\alpha_k^{(N)}}$
- repeat

7. The KL Divergence

The KL Divergence between two discrete distributions $p(x = k) = p_k$ and $p(x = k) = q_k$ is defined as $\mathcal{KL}(q, p) = \sum_{k=1}^{K} q_k \log \frac{q_k}{p_k}$.

(a) Prove that the KL Divergence is non-negative and that it attains its unique minimum when $q_k = p_k$.

(b) A machine learner has a target distribution $\mathbf{p} = [p_1, p_2, p_3, p_4, p_5, p_6] = [1, 1, 0, 0, 1, 1]/4$ which they want to fit with approximating distribution q. The choices for q available to the machine learner are: $q_1 = [1, 1, 0, 0, 0, 0]/2$, $q_2 = [0, 1, 1, 0, 0, 0]/2$, $q_3 = [0, 0, 1, 1, 0, 0]/2$, $q_4 = [0, 0, 0, 1, 1, 0]/2$, $q_5 = [0, 0, 0, 0, 1, 1]/2$, and $q_6 = [1, 1, 1, 1, 1, 1]/6$.

i. Determine the distribution(s) q_i that minimises $\mathcal{KL}(q_i, p)$. Comment on your result.

ii. Determine the distribution(s) q_i that minimises $\mathcal{KL}(p, q_i)$. Comment on your result.

Note that, by convention, $0 \times \log 0 = 0$ since $\lim_{\Delta \to 0} \Delta \log \Delta = 0$.

a) $KL(q \| p) = \sum_k q_k \log \frac{q_k}{p_k}$

constraint that probabilities sum to 1

$0 = \frac{\partial}{\partial q_l} \left[\sum_k q_k \log \frac{q_k}{p_k} + \lambda \left(\sum_k q_k - 1 \right) \right]$

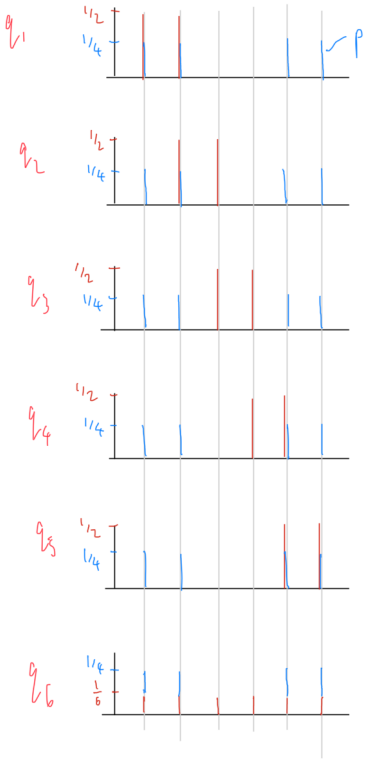
Lagrange multiplier

$0 = \log \frac{q_l}{p_l} + \frac{q_l}{q_l} + \lambda$

$\Rightarrow q_l = p_l$ @which $KL(q^{opt} \| p) = \sum_k p_k \log \frac{p_k}{p_k} = 0$

$\frac{\partial^2}{\partial q_l^2} KL(q \| p) = \frac{1}{q_l} > 0 \Rightarrow$ maximum

b)



	$KL(q \| p)$	$KL(p \| q)$
q_1	$\log 2$	∞
q_2	∞	∞
q_3	∞	∞
q_4	∞	∞
q_5	$\log 2$	∞
q_6	∞	$\log \frac{3}{2}$

$KL(q||p)$ is zero avoiding & tends to fit single modes of p with q

$KL(p||q)$ is not zero avoiding & tends to cover multiple modes of p with q

8. Mixtures of Gaussians and EM*

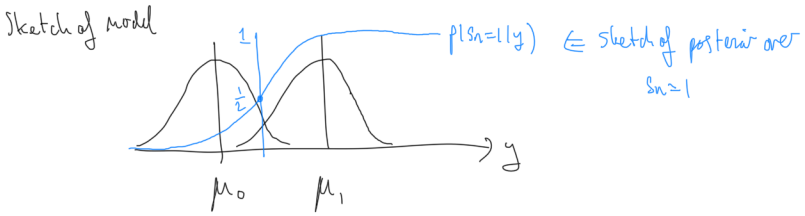
A set of N scalar data points $\{y_n\}_{n=1}^N$ are modelled using a mixture of Gaussians containing two equiprobable components with unknown means (μ_0 and μ_1) and unit variances,

$$p(s_n = 1) = \frac{1}{2}, \quad p(y_n|s_n = 0) = \mathcal{N}(y_n; \mu_0, 1), \quad p(y_n|s_n = 1) = \mathcal{N}(y_n; \mu_1, 1).$$

(a) Compute the posterior distribution over the components, $p(s_n = 1|y_n)$ and sketch how this varies as a function of the observed data y_n. Briefly discuss how this relates to logistic classification.

(b) Explain how your solution to (a) can be used in the EM algorithm to estimate the component means. Your answer should include a expression for the M-step update.

(c) Do you expect the EM algorithm to overfit when used to train this model?



a) $$p(s_n=1|y_n) = \frac{p(s_n=1, y_n)}{p(s_n=1, y_n) + p(s_n=0, y_n)} = \frac{1}{1 + \frac{p(s_n=0, y_n)}{p(s_n=1, y_n)}}$$

$$= \frac{1}{1 + \exp\left(-\frac{1}{2}(y_n-\mu_0)^2 + \frac{1}{2}(y_n-\mu_1)^2\right)}$$

$$= \frac{1}{1 + \exp\left(-\left[y_n(\mu_1-\mu_0) + \frac{\mu_1^2}{2} - \frac{\mu_0^2}{2}\right]\right)}$$

$$= \frac{1}{1+\exp(-\alpha)} = \text{logistic function}$$

Just like logistic regression with weights $(\mu_1 - \mu_0)$ & bias $+\frac{\mu_1^2}{2} - \frac{\mu_0^2}{2}$

b) E-Step computes $p(s_n = 1 | y_n)$ using expression above

M-Step

$$\mu_1^{(new)} = \frac{\sum_n p(s_n = 1 | y_n)\, y_n}{\sum_n p(s_n = 1 | y_n)}$$

mean of data each weighted by probability they come from class 1

$$\mu_0^{(new)} = \frac{\sum_n (1 - p(s_n = 1 | y_n))\, y_n}{\sum_n (1 - p(s_n = 1 | y_n))}$$

$p(s_n = 0 | y_n)$

as above, but weighted by prob come from class 0

c) Overfitting in a MoG model can occur when variances shrink to zero. As the variances are fixed to unity in this model over fitting is less likely.

The EM Algorithm

9. Factor Analysis and EM*

The noisy depth sensor from question 2 in example sheet 1 is used to collect measurements of the distances to a set of N objects that are unknown distances d_n metres away. The object depths can be assumed, *a priori*, to be distributed according to independent standard Gaussian distributions $p(d_n) = \mathcal{N}(d_n; 0, 1)$. As before, the depth sensor returns y_n a noisy measurement of the depth, that is also assumed to be Gaussian $p(y_n|d_n, \sigma_y^2) = \mathcal{N}(y_n; d_n, \sigma_y^2)$.

The variance of the σ_y^2 sensor noise is unknown and must be estimated from the measured depths $\{y_n\}_{n=1}^N$ using maximum likelihood.

(a) Derive the steps of the EM algorithm for performing this. Your answer should include the E-Step (you may find your solution to question 2 in example sheet 1 useful here) and the M-Step.

(b) An alternative approach to maximum-likelihood learning would optimise the log-likelihood $p(\{y_n\}_{n=1}^N|\sigma_y^2)$ directly. Compute the objective function for this procedure. How do you think this approach will perform compared to EM?

$$p(d_n) = \mathcal{N}(d_n; 0, 1)$$

$$p(y_n | d_n, \sigma_y^2) = \mathcal{N}(y_n; d_n, \sigma_y^2)$$

in E-Step set $q_n(d_n) = p(d_n | y_n, \sigma_y^2) = \mathcal{N}(d_n; \mu_{d_n|y_n}, \sigma^2_{d_n|y_n})$

where $\mu_{d_n|y_n} = \frac{y_n}{1+\sigma_y^2}$ $\quad \sigma^2_{d_n|y_n} = \frac{\sigma_y^2}{1+\sigma_y^2}$ (See Q2 Examples Sheet 1)

in M-Step

$$\sigma_y^2 = \underset{\sigma_y^2}{\operatorname{argmax}} \underbrace{\mathbb{E}_q\left[\log p(\{d_n\}_{n=1}^N, \{y_n\}_{n=1}^N | \sigma_y^2) \right]}_{G(\sigma_y^2)}$$

↑ "average log-joint"

$$G(\sigma_y^2) = \mathbb{E}_q\left[\sum_n \log p(d_n) + \sum_n \log p(y_n | d_n, \sigma_y^2) \right]$$

$$= \underbrace{c}_{\text{constant wrt } \sigma_y^2} - \frac{1}{2\sigma_y^2} \sum_n \mathbb{E}_{q(d_n)}\left[(y_n - d_n)^2 \right] - \frac{N}{2} \log 2\pi\sigma_y^2 \quad ★$$

$$= c - \frac{1}{2\sigma_y^2} \sum_n \left(y_n^2 + \mathbb{E}_{q(d_n)}[d_n^2] - 2y_n \mathbb{E}_{q(d_n)}[d_n] \right) - \frac{N}{2} \log 2\pi\sigma_y^2$$

$$\frac{dG(\sigma_y^2)}{d\sigma_y^2} = \frac{1}{2\sigma_y^4} \sum_n \left(y_n^2 + \mathbb{E}_{q(d_n)}[d_n^2] - 2y_n \mathbb{E}_{q(d_n)}[d_n] \right) - \frac{N}{2\sigma_y^2} = 0$$

$$\therefore \sigma_y^2 = \frac{1}{N}\sum_n y_n^2 + \frac{1}{N}\sum_n \left(\mu_{d_n|y_n}^2 + \sigma_{d_n|y_n}^2\right) - \frac{2}{N}\sum_n y_n \mu_{d_n|y_n}$$

Algorithmically this is what you need, although for intuition ✡ yields:

$$\sigma_y^2 = \frac{1}{N}\sum_n \mathbb{E}_{q(d_n)}\left[(y_n - d_n)^2\right]$$

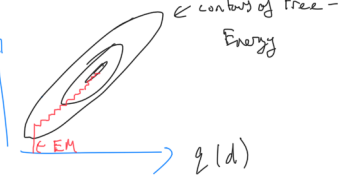
i.e. the noise variance is the average (over data points n & the posterior) of the squared error between y_n & d_n.

b) $$p(y_n|\sigma_y^2) = \mathcal{N}(y_n; 0, 1+\sigma_y^2)$$

$$\therefore \log p(\{y_n\}_{n=1}^N|\sigma_y^2) = -\frac{N}{2}\log[2\pi(1+\sigma_y^2)] - \frac{1}{2(1+\sigma_y^2)}\sum_n y_n^2$$

Optimising this function for σ_y^2 (eg by gradient ascent) would be MUCH faster than EM. EM performs co-ordinate ascent of the free-energy in q & θ which tends to be slow.

It turns out that if EM is analytic, so too is direct optimisation of the log likelihood which is almost always to be preferred.



10. Gaussian Mixture Models and EM** (beyond tripos standard; optional)

A Gaussian Mixture Model for D-dimensional data $\{\boldsymbol{x}_n\}_{n=1}^N$ comprises a categorical distribution over the class membership variables $p(s_n = k|\theta) = \pi_k$ and a general multivariate Gaussian distribution over the observed data given the class membership variables, $p(\boldsymbol{x}_n|s_n = k, \theta) = \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{m}_k, \Sigma_k)$, (i.e. Σ_k is not isotopic). The posterior distribution over the class membership variables has been computed $p(s_n = k|\boldsymbol{x}_n, \theta) = q_{n,k}$.

(a) Derive the M-Step update of the EM algorithm, i.e. the formulae for updating the parameters $\{\pi_k, \boldsymbol{m}_k, \Sigma_k\}_{k=1}^K$ using the posterior probabilities, $q_{n,k}$.

(b) A friend suggests that it might be possible to speed up the EM algorithm by updating the posterior distribution of only a subset of $K < N$ data points during the E-Step that are selected uniformly at random each time, and then updating the parameters using the same expressions derived in part (a).

i. Will this procedure converge to a (local) optimum of the likelihood?

ii. Will this partial E-Step update procedure be computationally more efficient?

You may find the following identities useful,

$$\frac{\mathrm{d}}{\mathrm{d}\alpha} \log\det(\Sigma) = \mathrm{trace}\left(\Sigma^{-1}\frac{\mathrm{d}\Sigma}{\mathrm{d}\alpha}\right), \quad \frac{\mathrm{d}}{\mathrm{d}\alpha}\Sigma^{-1} = -\Sigma^{-1}\frac{\mathrm{d}\Sigma}{\mathrm{d}\alpha}\Sigma^{-1}.$$

$$\mathcal{F}(q(\underline{S}), \theta) = \sum_n q(S_n) \log \frac{p(\underline{x}_n | S_n, \theta) p(S_n | \theta)}{q(S_n)}$$

$$q(S_n = k) = q_{nk}$$

$$\therefore \quad \mathcal{F}(\underline{q}(\underline{S}), \theta) = \sum_n \sum_k q_{nk} \log \frac{p(\underline{x}_n | S_n = k, \theta) p(S_n = k | \theta)}{q(S_n = k)}$$

where

$$p(\underline{x}_n | S_n = k, \theta) = N(\underline{x}_n ; \underline{\mu}_k, \underline{\underline{\Sigma}}_k) \qquad p(S_n = k | \theta) = \pi_k$$

$$\therefore \quad \mathcal{F}(\underline{q}(\underline{S}), \theta) = \sum_n \sum_k q_{nk} \left[-\frac{1}{2} \log \det(2\pi \underline{\underline{\Sigma}}_k) - \frac{1}{2} (\underline{x}_n - \underline{\mu}_k)^T \underline{\underline{\Sigma}}_k^{-1} (\underline{x}_n - \underline{\mu}_k) + \log \pi_k \right] + c$$

(c: does not depend on parameters)

$$\frac{\partial \mathcal{F}}{\partial \mu_{k'd'}} = \frac{\partial}{\partial \mu_{k'd'}} \left[-\frac{1}{2} \sum_n \sum_k q_{nk} \sum_{ij} (x_{in} - \mu_{ki}) \Sigma^{-1}_{kij} (x_{jn} - \mu_{kj}) \right] = 0$$

$$= \frac{1}{2} \sum_n q_{nk'} \left[\sum_j \Sigma^{-1}_{k'd'j} (x_{jn} - \mu_{k'j}) + \sum_i (x_{in} - \mu_{ki}) \Sigma^{-1}_{kid'} \right]$$

$$= \sum_n q_{nk'} \sum_j \Sigma^{-1}_{k'd'j} (x_{jn} - \mu_{k'j}) = \left[\Sigma^{-1}_{k'} \sum_n q_{nk'} (\underline{x}_n - \underline{\mu}_{k'}) \right]_{d'}$$

$$\therefore \underline{\mu}_k = \frac{\sum_n q_{kn}\underline{x}_n}{\sum_n q_{kn}}$$

alternatively, you could differentiate wrt $\Sigma^{-1}_{k'i'j'}$ & use $\det(\Sigma) = \frac{1}{\det(\Sigma^{-1})}$

$$\frac{d\mathcal{F}}{d\Sigma_{k'i'j'}} = \frac{d}{d\Sigma_{k'i'j'}}\left[\sum_{n,k} q_{nk}\left[-\tfrac{1}{2}\log\det(2\pi\underline{\underline{\Sigma}}_k) - \tfrac{1}{2}(\underline{x}_n-\underline{\mu}_k)^T\underline{\underline{\Sigma}}^{-1}_k(\underline{x}_n-\underline{\mu}_k)\right]\right]$$

$$\frac{d}{d\alpha}\log\left(\det\underline{\underline{\Sigma}}_k\right) = \mathrm{trace}\left(\underline{\underline{\Sigma}}^{-1}_k\frac{d\underline{\underline{\Sigma}}_k}{d\alpha}\right) \qquad \frac{d}{d\alpha}\underline{\underline{\Sigma}}^{-1}_k = -\underline{\underline{\Sigma}}^{-1}_k\frac{d\underline{\underline{\Sigma}}}{d\alpha}\underline{\underline{\Sigma}}^{-1}_k$$

$$\frac{d\mathcal{F}}{d\Sigma_{k'i'j'}} = \sum_{nk} q_{nk}\left[-\tfrac{1}{2}\sum_{ij}\Sigma^{-1}_{kij}\frac{d\Sigma_{kji}}{d\Sigma_{k'i'j'}} + \tfrac{1}{2}\sum_{ijlm}(x_{ni}-\mu_{ki})\Sigma^{-1}_{kil}\frac{d\Sigma_{klm}}{d\Sigma_{k'i'j'}}\Sigma^{-1}_{kmj}(x_{nj}-\mu_{kj})\right]$$

$$= \sum_{nk} q_{nk}\left[-\frac{1}{2}\sum_{ij}\Sigma^{-1}_{kij}\,\delta_{kk'}\delta_{i'j}\delta_{j'i} + \frac{1}{2}\sum_{ijlm}(x_{ni}-\mu_{ki})\Sigma^{-1}_{kil}\,\delta_{k'k}\delta_{i'l}\delta_{mj'}\Sigma^{-1}_{kmj}(x_{nj}-\mu_{kj})\right]$$

$$= \frac{1}{2}\sum_n q_{nk'}\left(-\Sigma^{-1}_{k'j'i'} + \sum_{ij}(x_{ni}-\mu_{k'i})\Sigma^{-1}_{k'i'i}\Sigma^{-1}_{k'j'j}(x_{nj}-\mu_{k'j})\right)$$

$\Rightarrow \sum_n q_{nk'}\left(-\underline{\underline{\Sigma}}^{-1}_{k'} + \underline{\underline{\Sigma}}^{-1}_{k'}(\underline{x}_n-\underline{\mu}_{k'})(\underline{x}_n-\underline{\mu}_{k'})^T\underline{\underline{\Sigma}}^{-1}_{k'}\right) = 0$

$$0 = \sum_n q_{nk'}\left(-\underline{\underline{\Sigma}}^{-1}_{k'} + \underline{\underline{\Sigma}}^{-1}_{k'}(\underline{x}_n-\underline{\mu}_{k'})(\underline{x}_n-\underline{\mu}_{k'})^T\underline{\underline{\Sigma}}^{-1}_{k'}\right)$$

$$\Rightarrow \underline{\underline{\Sigma}}_{k'} = \sum_n q_{nk'}(\underline{x}_n-\mu_{k'})(\underline{x}_n-\mu_{k'})^T / \sum_n q_{nk'}$$

pre & post multiply by $\underline{\underline{\Sigma}}_{k'}$

To find π we need to use Lagrange multipliers:

$$\frac{d}{d\pi_{k'}}\left(\mathcal{F}(q,\theta) - \lambda\left(\sum_k \pi_k - 1\right)\right) = \frac{d}{d\pi_{k'}}\left[\sum_{nk}\left(q_{nk}\log\pi_k - \lambda\pi_k\right) - 1\right]$$

$$= \sum_n q_{nk'}\left(\frac{1}{\pi_{k'}} - \lambda\right) = 0$$

$$\Rightarrow \pi_{k'} = \frac{\sum_n q_{nk'}}{\sum_{nk} q_{nk'}}$$ ← from constraint that $\sum_k \pi_k = 1$

$$= \frac{1}{N}\sum_n q_{nk'}$$

b i) The free-energy is guaranteed to improve (or stay the same) if only a fraction of the datapoint posteriors are updated.
∴ the procedure will converge to a local optimum of the likelihood as for normal EM

ii) It's not completely clear that the procedure will help in all cases - for although the cost of the E-Step is reduced (from $O(N)$ to $O(K)$) convergence will take more steps in general. However, when there are lots of data, you don't need to visit all of them to figure out a good update for the parameters so this means the procedure is likely to be substantially more efficient in these cases.