Linear regression

Regression

Regression

- A regression problem is one that identifies patterns and regularities in the mapping btwn i/p variables $\underline{x}^{(n)}$ and a corresponding continuous o/p variable $y^{(n)}$ from a training data set $D=\{(\underline{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$ (Note $\underline{x}^{(n)} = [x_1^{(n)} \dots x_D^{(n)}]^T \in \mathbb{R}^D$ and $y_n \in \mathbb{R}$).
- A regression method typically returns a conditional distribution $p(y_*|\underline{x}_*)$ for the unknown y_* associated w/ a new test i/p $\underline{x}_*$ → useful for predictions / interpretations (find correlation)

↳ The characteristics of $p(y^{(n)}|\underline{x}^{(n)})$ should capture patterns observed in the data (e.g. changes in mean/variance)

- We use a model to encode, as function of some parameters $\underline{\theta}$, the type of patterns we expect to find in the data.

Linear regression model

- For the linear regression model, we assume that the relationship btwn i/ps $\underline{x}^{(n)}$ and o/ps $y^{(n)}$ in $p(y^{(n)}|\underline{x}^{(n)})$ is linear, w/ $\underline{\theta}$ as parameters specifying this conditional distribution.
- For the 1D case, we have $\underline{\theta} = \{\sigma^2, w_0, w_1\}$

$$y^{(n)} = w_0 + w_1 x^{(n)} + \varepsilon^{(n)}, \qquad \varepsilon^{(n)} \sim N(0, \sigma^2)$$

$y^{(n)}$ is the sum of a const. and a Gaussian RV → $y^{(n)}$ is Gaussian distributed.

$$E[y^{(n)}] = w_0 + w_1 x^{(n)} + \cancel{E[\varepsilon^{(n)}]}^{0} = w_0 + w_1 x^{(n)} \qquad Var[y^{(n)}] = E[(y^{(n)} - E[y^{(n)}])^2] = E[\varepsilon^{(n)2}] = \sigma^2$$

$$\therefore p(y^{(n)}|\underline{x}^{(n)}, \underline{\theta}) = N(y^{(n)}|w_0 + w_1 x^{(n)}, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2}\frac{(y^{(n)} - w_0 - w_1 x^{(n)})^2}{\sigma^2}\right]$$

- For higher dimensions, we have $\underline{\theta} = \{\sigma^2, \underline{w} = [w_0 \dots w_D]^T\}$

$$y^{(n)} = w_0 + w_1 x_1^{(n)} + \dots + w_D x_D^{(n)} = [w_0 \dots w_D] \begin{bmatrix} x_0^{(n)} \\ \vdots \\ x_D^{(n)} \end{bmatrix} + \varepsilon_n = \underline{w}^T \underline{\tilde{x}}^{(n)} + \varepsilon_n, \qquad \varepsilon_n \sim N(0, \sigma^2)$$

where $\underline{\tilde{x}} = [1, \underline{x}_n]^T$ is the augmented i/p

$y^{(n)}$ is the sum of a const. and a Gaussian RV → $y^{(n)}$ is Gaussian distributed.

$$E[y^{(n)}] = \underline{w}^T \underline{\tilde{x}}^{(n)} + \cancel{E[\varepsilon^{(n)}]}^{0} = \underline{w}^T \underline{\tilde{x}}^{(n)} \qquad Var[y^{(n)}] = E[(y^{(n)} - E[y^{(n)}])^2] = E[\varepsilon^{(n)2}] = \sigma^2$$

$$\therefore p(y^{(n)}|\underline{x}^{(n)}, \underline{\theta}) = N(y^{(n)}|\underline{w}^T \underline{\tilde{x}}^{(n)}, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2}\frac{(y^{(n)} - \underline{w}^T \underline{\tilde{x}}^{(n)})^2}{\sigma^2}\right]$$

Maximum likelihood estimate (MLE).

- We find the MLE for $\underline{\theta}$ by max. the likelihood function $p(y^{(1)} \dots y^{(N)}|\underline{x}^{(1)} \dots \underline{x}^{(N)}, \underline{\theta})$ wrt $\underline{\theta}$.

$$\underline{\theta}^{ML} = \arg\max_{\underline{\theta}} p(y^{(1)} \dots y^{(N)}|\underline{x}^{(1)} \dots \underline{x}^{(N)}, \underline{\theta})$$

- In practice, we max. the log-likelihood function $L(\underline{\theta}) = \log p(y^{(1)} \dots y^{(N)}|\underline{x}^{(1)} \dots \underline{x}^{(N)}, \underline{\theta})$

↳ we get the same soln since logarithm is monotonically increasing

$$\underline{\theta}^{ML} = \arg\max_{\underline{\theta}} p(y^{(1)} \dots y^{(N)}|\underline{x}^{(1)} \dots \underline{x}^{(N)}, \underline{\theta}) = \arg\max_{\underline{\theta}} L(\underline{\theta})$$

↳ taking the log can prevent overflow/underflow problems when computing products.

- Assuming that each data pt is independent, then

$$p(\underline{y} \mid \underline{X}, \underline{\theta}) = p(y^{(1)} \dots y^{(N)} \mid \underline{x}^{(1)} \dots \underline{x}^{(N)}, \underline{\theta}) = \prod_{n=1}^{N} p(y^{(n)} \mid \underline{x}^{(n)}, \underline{\theta}) = \prod_{n=1}^{N} \mathcal{N}(y^{(n)} \mid \underline{w}^T \underline{x}^{(n)}, \sigma^2)$$

the log likelihood $L(\underline{\theta})$ is thus given by

$$L(\underline{\theta}) = \log p(y^{(1)} \dots y^{(N)} \mid \underline{x}^{(1)} \dots \underline{x}^{(N)}, \underline{\theta}) = \sum_{n=1}^{N} \log p(y^{(n)} \mid \underline{x}^{(n)}, \underline{\theta}) = \sum_{n=1}^{N} \log \mathcal{N}(y^{(n)} \mid \underline{w}^T \underline{x}^{(n)}, \sigma^2)$$

$$= \sum_{n=1}^{N} \left[-\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(y^{(n)} - \underline{w}^T \underline{x}^{(n)})^2 \right] = -\frac{N}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(\underline{y} - \underline{\Phi}\underline{w})^T(\underline{y} - \underline{\Phi}\underline{w})$$

$$\boxed{L(\underline{\theta}) = -\frac{N}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\left(\underline{y}^T\underline{y} + \underline{w}^T\underline{\Phi}^T\underline{\Phi}\underline{w} - \underline{w}^T\underline{\Phi}^T\underline{y} - \underline{y}^T\underline{\Phi}\underline{w}\right)}$$

where $\underline{y} = [y_1 \dots y_N]^T$ and $\underline{\Phi} = \begin{bmatrix} \leftarrow & \underline{x}^{(1)T} & \rightarrow \\ & \vdots & \\ \leftarrow & \underline{x}^{(N)T} & \rightarrow \end{bmatrix}$

- Differentiating $L(\underline{\theta})$ wrt $\underline{w}$,

$$\frac{\partial L(\underline{\theta})}{\partial \underline{w}} = \frac{\partial}{\partial \underline{w}}\left(-\frac{1}{2\sigma^2}\left(\underline{w}^T\underline{\Phi}^T\underline{\Phi}\underline{w} - \underline{w}^T\underline{\Phi}^T\underline{y} - \underline{y}^T\underline{\Phi}\underline{w}\right)\right) = -\frac{1}{2\sigma^2}\left(2\underline{\Phi}^T\underline{\Phi}\underline{w} - 2\underline{\Phi}^T\underline{y}\right) = 0$$

$$\therefore \boxed{\underline{w} = (\underline{\Phi}^T\underline{\Phi})^{-1}\underline{\Phi}^T\underline{y}}$$

Differentiating $L(\underline{\theta})$ wrt σ^2,

$$\frac{\partial L(\underline{\theta})}{\partial \sigma^2} = \frac{\partial}{\partial \sigma^2}\left(-\frac{N}{2}\log(2\pi\sigma^2)\right) - \frac{1}{2\sigma^2}(\underline{w}^T\underline{\Phi}^T - \underline{y}^T)(\underline{\Phi}\underline{w} - \underline{y}) = -\frac{N}{2\sigma^2} + \frac{1}{2\sigma^4}(\underline{\Phi}\underline{w} - \underline{y})^T(\underline{\Phi}\underline{w} - \underline{y}) = 0$$

$$\therefore \boxed{\sigma^2 = \frac{1}{N}(\underline{\Phi}\underline{w} - \underline{y})^T(\underline{\Phi}\underline{w} - \underline{y})}$$

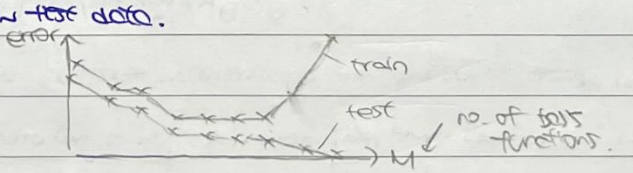
- when $N \leq D$, the MLE $\underline{w} = (\underline{\Phi}^T\underline{\Phi})^{-1}\underline{\Phi}^T\underline{y}$ is not defined, as $\underline{\Phi}^T\underline{\Phi}$ is not invertible
  → we have many values of $\underline{w}$ that fit the training data equally well w/ zero error.

Non-linear (basis function) regression

- Linear regression can model non-linear relationships by replacing $\underline{x}^{(n)}$ w/ some non-linear function of the i/ps $\underline{\phi}(\underline{x}^{(n)}) = [\phi_1(\underline{x}^{(n)}) \dots \phi_M(\underline{x}^{(n)})]^T$ (so $\underline{w}^T\underline{x}^{(n)} \rightarrow \underline{w}^T\underline{\phi}(\underline{x}^{(n)})$)
- Inference does not change, we just replace each $\underline{x}^{(n)}$ w/ the new $\underline{\phi}(\underline{x}^{(n)})$
- common basis functions $\phi_m(\underline{x})$ include:
  - ↳ polynomial for 1D data: $\phi_m(\underline{x}^{(n)}) = x^{(n)m}$
  - ↳ Gaussian radial basis functions: $\phi_m(\underline{x}^{(n)}) = \exp\left[-\frac{(\underline{x}^{(n)} - \underline{c}_m)^T(\underline{x}^{(n)} - \underline{c}_m)}{2s^2}\right]$ (centre $\underline{c}_m$, width s^2)
  - ↳ sigmoidal basis functions: $\phi_m(\underline{x}^{(n)}) = \sigma\left(\frac{(\underline{x}^{(n)} - \underline{c}_m)^T(\underline{x}^{(n)} - \underline{c}_m)}{s^2}\right)$, $\sigma(x) = \frac{1}{1+\exp(x)}$
- Linear in terms of transformed i/ps $\underline{\phi}(\underline{x}^{(n)})$ but non-linear in terms of original i/ps $\underline{x}^{(n)}$.

overfitting.

- A large no. of basis functions can lead to overfitting – the model fits the training data well but it performs poorly on new test data.



- As M increases, the magnitude of the entries of the MLE of $\underline{w}$ become large → large oscillations in the model's predictions → solve this by using a prior to favour the entries of $\underline{w}$ to be small.

Bayesian linear regression

Multivariate Gaussian distribution

- The density of a D-dimensional vector $\underline{x} \in \mathbb{R}^D$ is

$$\mathcal{N}(\underline{x} \mid \underline{m}, \underline{\underline{V}}) = \frac{1}{\sqrt{(2\pi)^D |\underline{\underline{V}}|}} \exp\left[-\frac{1}{2}(\underline{x}-\underline{m})^T \underline{\underline{V}}^{-1}(\underline{x}-\underline{m})\right] \propto \exp\left[-\frac{1}{2}\underline{x}^T \underline{\underline{V}}^{-1}\underline{x} + \underline{m}^T \underline{\underline{V}}^{-1}\underline{x}\right]$$

i.e. the density is prop. to the exponential of a quadratic function of $\underline{x}$, w/ normalization const. Z

$$Z = \sqrt{(2\pi)^D |\underline{\underline{V}}|} \exp\left(\frac{1}{2}\underline{m}^T \underline{\underline{V}}^{-1}\underline{m}\right)$$

where $\underline{m} \in \mathbb{R}^D$ is the mean vector and $\underline{\underline{V}} \in \mathbb{R}^{D \times D}$ is the covariance matrix

- The general Gaussian $p(\underline{x}) = \mathcal{N}(\underline{x} \mid \underline{m}, \underline{\underline{V}})$ can be obtained from the standard Gaussian $p(\underline{z}) = \mathcal{N}(\underline{z} \mid \underline{0}, \underline{\underline{I}})$

$$\underline{x} = \underline{m} + \underline{\underline{V}}^{1/2}\underline{z}$$

where $\underline{\underline{V}}^{1/2}$ is the matrix square root, defined as $\underline{\underline{V}} = \underline{\underline{V}}^{1/2}(\underline{\underline{V}}^{1/2})^T$ ← Cholesky decomposition

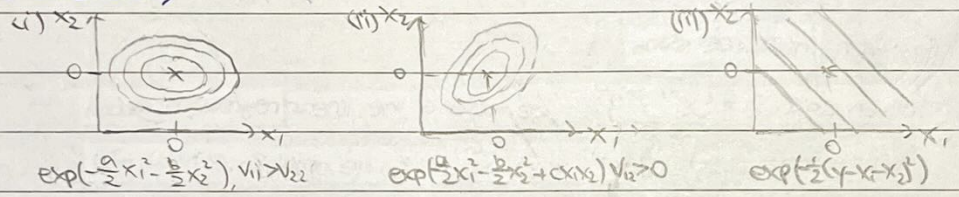
Effect of parameters on the density function

- For the general Gaussian density,

$$p(\underline{x}) = \mathcal{N}(\underline{x} \mid \underline{m}, \underline{\underline{V}}) = \frac{1}{\sqrt{(2\pi)^D |\underline{\underline{V}}|}} \exp\left[-\frac{1}{2}(\underline{x}-\underline{m})^T \underline{\underline{V}}^{-1}(\underline{x}-\underline{m})\right]$$

the parameter $\underline{m}$ determines the mode location and $\underline{\underline{V}}$ scales + rotates the space.

- For the $D=2$ case,



(i) $\exp(-\frac{a}{2}x_1^2 - \frac{b}{2}x_2^2) = c \rightarrow ax_1^2 + bx_2^2 = k$

(ii) $\exp(-\frac{a}{2}x_1^2 - \frac{b}{2}x_2^2 + cx_1x_2) = c \rightarrow ax_1^2 + bx_2^2 - 2cx_1x_2 = k$

(iii) $\exp(-\frac{1}{2}(y - x_1 - x_2)^2) = c \rightarrow x_1 + x_2 = y + k$

Linear combination of Gaussian RVs

- Let $p(\underline{x}) = \mathcal{N}(\underline{x} \mid \underline{0}, \underline{\underline{V}}_1)$ and $p(\underline{e}) = \mathcal{N}(\underline{e} \mid \underline{0}, \underline{\underline{V}}_2)$. Consider the transformation

$$\underline{y} = \underline{\underline{M}}\underline{x} + \underline{e}$$

As linear combinations of independent Gaussian RVs are Gaussian, $p(\underline{y})$ is Gaussian, $p(\underline{y}) = \mathcal{N}(\underline{y} \mid \underline{m}_y, \underline{\underline{V}}_y)$

↳ $\underline{m}_y = E[\underline{y}] = \underline{\underline{M}}E[\underline{x}] + E[\underline{e}] = 0 \rightarrow \boxed{\underline{m}_y = \underline{0}}$

↳ $\underline{\underline{V}}_y = E[\underline{y}\underline{y}^T] - E[\underline{y}]E[\underline{y}]^T = E[(\underline{e} + \underline{\underline{M}}\underline{x})(\underline{e} + \underline{\underline{M}}\underline{x})^T] = E[\underline{e}\underline{e}^T + \underline{e}\underline{x}^T\underline{\underline{M}}^T + \underline{\underline{M}}\underline{x}\underline{e}^T + \underline{\underline{M}}\underline{x}\underline{x}^T\underline{\underline{M}}^T]$

$= E[\underline{e}\underline{e}^T] + E[\underline{e}]E[\underline{x}]^T\underline{\underline{M}}^T + \underline{\underline{M}}E[\underline{x}]E[\underline{e}]^T + \underline{\underline{M}}E[\underline{x}\underline{x}^T]\underline{\underline{M}}^T = \underline{\underline{V}}_2 + \underline{\underline{M}}\underline{\underline{V}}_1\underline{\underline{M}}^T \rightarrow \boxed{\underline{\underline{V}}_y = \underline{\underline{V}}_2 + \underline{\underline{M}}\underline{\underline{V}}_1\underline{\underline{M}}^T}$

* Note that $\underline{x}$ and $\underline{e}$ are assumed to be independent, so $E[\underline{x}\underline{e}^T] = E[\underline{x}]E[\underline{e}]^T$ etc.

Product of Gaussian densities

- Let $p(\underline{x}) = N(\underline{x}|\underline{m}_1, \underline{V}_1)$ and $q(\underline{x}) = N(\underline{x}|\underline{m}_2, \underline{V}_2)$, consider the product

$$f(\underline{x}) \propto p(\underline{x})\, q(\underline{x})$$

since the product of exponentials of quadratic functions is also the exponential of a quadratic function, $f(\underline{x})$ is also Gaussian, $f(\underline{x}) = N(\underline{x}|\underline{m}_3, \underline{V}_3)$

$$f(\underline{x}) \propto p(\underline{x})q(\underline{x}) = N(\underline{x}|\underline{m}_1, \underline{V}_1)N(\underline{x}|\underline{m}_2, \underline{V}_2) = \frac{1}{\sqrt{|2\pi \underline{V}_1|}} \frac{1}{\sqrt{|2\pi \underline{V}_2|}} \exp\left[-\tfrac{1}{2}(\underline{x}-\underline{m}_1)^T \underline{V}_1^{-1}(\underline{x}-\underline{m}_1) - \tfrac{1}{2}(\underline{x}-\underline{m}_2)^T \underline{V}_2^{-1}(\underline{x}-\underline{m}_2)\right]$$

$$\propto \exp\left[-\tfrac{1}{2}\underline{x}^T(\underline{V}_1^{-1}+\underline{V}_2^{-1})\underline{x} + (\underline{m}_1^T\underline{V}_1^{-1} + \underline{m}_2^T\underline{V}_2^{-1})\underline{x}\right]$$

Noting that $f(\underline{x}) = N(\underline{x}|\underline{m}_3, \underline{V}_3) \propto \exp\left(-\tfrac{1}{2}\underline{x}^T\underline{V}_3^{-1}\underline{x} + \underline{m}_3^T\underline{V}_3^{-1}\underline{x}\right)$,

$\underline{V}^{-1}, \underline{m}^T\underline{V}^{-1}$ are "natural parameters" → simply sum

$$\underline{V}_3^{-1} = (\underline{V}_1^{-1} + \underline{V}_2^{-1}) \quad \rightarrow \quad \underline{V}_3 = (\underline{V}_1^{-1} + \underline{V}_2^{-1})^{-1}$$

$$\underline{m}_3^T\underline{V}_3^{-1} = \underline{m}_1^T\underline{V}_1^{-1} + \underline{m}_2^T\underline{V}_2^{-1} \quad \rightarrow \quad \underline{m}_3 = \underline{V}_3(\underline{m}_1^T\underline{V}_1^{-1} + \underline{m}_2^T\underline{V}_2^{-1})^T$$

Completing the square.

- Let $p(\underline{x}) = N(\underline{x}|\underline{m}, \underline{V}) \propto \exp\left[-\tfrac{1}{2}\underline{x}^T\underline{V}^{-1}\underline{x} + \underline{m}^T\underline{V}^{-1}\underline{x}\right]$ and we are given an exponential of a quadratic function $q(\underline{x}) \propto \exp\left[-\tfrac{1}{2}\underline{x}^T\underline{B}\underline{x} + \underline{b}^T\underline{x}\right]$ to match p.

$$\underline{B} = \underline{V}^{-1} \rightarrow \underline{V} = \underline{B}^{-1}, \qquad \underline{b}^T = \underline{m}^T\underline{V}^{-1} \rightarrow \underline{m} = \underline{V}\underline{b}$$

Bayesian inference

- using a large no. of basis functions can lead to overfitting — the model fits the training data well but performs poorly on new test data → $\|\underline{\theta}^{ML}\|$ becomes large → large oscillations in model predictions.
- we can favour smooth solns using priors to enforce $\underline{\theta}$ to be small.
- The posterior distribution for $\underline{\theta}$, $p(\underline{\theta}|\underline{Y}, \underline{X})$ is obtained by Bayes rule

$$p(\underline{\theta}|\underline{Y}, \underline{X}) = \frac{p(\underline{Y}|\underline{X}, \underline{\theta})p(\underline{\theta}|\underline{X})}{p(\underline{Y}|\underline{X})}$$

where $p(\underline{Y}|\underline{X}, \underline{\theta}) = \prod_{n=1}^{N} p(y^{(n)}|\underline{x}^{(n)}, \underline{\theta})$ is the likelihood, $p(\underline{Y}|\underline{X})$ is the model evidence and $p(\underline{\theta}|\underline{X}) = p(\underline{\theta})$ is the prior distribution.

- The predictive distribution for y_* given a new corresponding $\underline{x}_*$ is

$$p(y_*|\underline{x}_*, \underline{Y}, \underline{X}) = \int p(y_*, \underline{\theta}|\underline{x}_*, \underline{Y}, \underline{X})\, d\underline{\theta} = \int p(y_*|\underline{x}_*, \underline{\theta})\, p(\underline{\theta}|\underline{Y}, \underline{X})\, d\underline{\theta}$$

i.e. weighted average of the predictions $p(y_*|\underline{x}_*, \underline{\theta})$, over the posterior distribution $p(\underline{\theta}|\underline{Y}, \underline{X})$.

- Exact inference is possible if prior and noise distributions are Gaussian.

Bayesian linear regression

- Given data $D = \{(\underline{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$, we assume the linear regression model

$$y^{(n)} = \underline{w}^T \underline{x}^{(n)} + \varepsilon_n \quad , \quad \varepsilon_n \sim N(0, \sigma^2)$$

To simplify inference, assume σ^2 is known — the only unknown is $\underline{w}$.

- Choose the prior for $\underline{w}$ to be a zero-mean isotropic Gaussian, precision matrix

$$p(\underline{w}) = N(\underline{w} | 0, \lambda^{-1} \underline{\underline{I}}) \propto \exp\left[-\tfrac{1}{2} \underline{w}^T \lambda \underline{\underline{I}} \underline{w}\right]$$

Recall that the likelihood under Gaussian noise is

$$p(y | \underline{X}, \underline{w}) = \prod_{n=1}^{N} N(y^{(n)} | \underline{w}^T \underline{x}^{(n)}, \sigma^2) \propto \exp\left[-\frac{\underline{w}^T \underline{X}^T \underline{X} \underline{w}}{2\sigma^2} + \frac{\underline{w}^T \underline{X}^T y}{\sigma^2}\right]$$

The posterior density is the product of two Gaussian densities → posterior is Gaussian $N(\underline{m}, \underline{\underline{V}})$

$$p(\underline{w} | y, \underline{X}, \sigma^2) \propto p(y | \underline{X}, \underline{w}) p(\underline{w}) \propto \exp\left[-\tfrac{1}{2} \underline{w}^T \underbrace{\left(\frac{\underline{\underline{X}}^T \underline{\underline{X}}}{\sigma^2} + \lambda \underline{\underline{I}}\right)}_{\underline{\underline{V}}^{-1}} \underline{w} + \underline{w}^T \underbrace{\frac{\underline{\underline{X}}^T y}{\sigma^2}}_{\underline{\underline{V}}^{-1} \underline{m}}\right]$$

Therefore $\underline{\underline{V}} = \left(\frac{\underline{\underline{X}}^T \underline{\underline{X}}}{\sigma^2} + \lambda \underline{\underline{I}}\right)^{-1}$, $\underline{m} = \underline{\underline{V}} \frac{\underline{\underline{X}}^T y}{\sigma^2}$

(Alternatively, note $\underline{\underline{V}}^{-1} = \frac{\underline{\underline{X}}^T \underline{\underline{X}}}{\sigma^2} + \left(\lambda \underline{\underline{I}}\right)^{-1}$, $\underline{\underline{V}}^T \underline{\underline{V}}^{-1} = \frac{\underline{\underline{X}}^T}{\sigma^2} + \underline{0}^T \left(\lambda \underline{\underline{I}}\right)^{-1}$)

- As $\lambda \to 0$ (prior is flat), $\underline{w}^{MAP} = \underline{m} = \underline{w}^{ML}$ i.e. ML and MAP solns are identical.

Bayesian predictive distribution

- The predictive distribution for y_* given a new corresponding $\underline{x}_*$ is

$$p(y_* | \underline{x}_*, \underline{X}, \underline{y}) = \int p(y_* | \underline{x}_*, \underline{w}) p(\underline{w} | \underline{X}, \underline{y}) d\underline{w} = \int N(y_* | \underline{x}_*^T \underline{w}, \sigma^2) N(\underline{w} | \underline{m}, \underline{\underline{V}}) d\underline{w}$$

- Instead of directly computing the integral, we have $y_* = \underline{w}^T \underline{x}_* + \varepsilon$, where $\underline{w} \sim N(\underline{m}, \underline{\underline{V}})$, $\varepsilon \sim N(0, \sigma^2)$,

$$p(y_* | \underline{x}_*, \underline{X}, \underline{y}) = N(y_* | m_*, V_*)$$

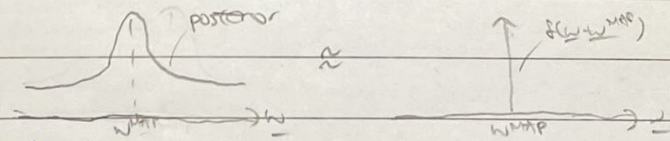
where $m_* = E[y_*] = E[\underline{w}^T \underline{x}_*] + E[\varepsilon] = \underline{m}^T \underline{x}_* \quad \rightarrow \boxed{m_* = \underline{m}^T \underline{x}_*}$

$V_* = \mathrm{Var}[y_*] = \mathrm{Var}[\underline{w}^T \underline{x}_*] + \mathrm{Var}[\varepsilon] = \underline{x}_*^T \underline{\underline{V}} \underline{x}_* + \sigma^2 \quad \rightarrow \boxed{V_* = \underline{x}_*^T \underline{\underline{V}} \underline{x}_* + \sigma^2}$

- We can reduce overfitting and obtain confidence bands $m_* \pm \frac{1}{2} V_*$ in our predictions.

Maximum a posteriori (MAP) inference

- Assume that the posterior is well approximated by a pt mass at its mode



The predictive distribution becomes

$$p(y_* | \underline{x}_*, \underline{X}, \underline{y}) = \int p(y_* | \underline{x}_*, \underline{w}) p(\underline{w} | \underline{X}, \underline{y}) d\underline{w} \approx \int p(y_* | \underline{x}_*, \underline{w}) \delta(\underline{w} - \underline{w}^{MAP}) d\underline{w} = p(y_* | \underline{x}_*, \underline{w}^{MAP})$$

- MAP inference fails to generate confidence bands in the resulting predictions.

- The MAP soln $\underline{w}^{MAP}$ is obtained as follows

$$\underline{w}^{MAP} = \arg\max_{\underline{w}} p(\underline{w} | \underline{X}, \underline{y}) = \arg\max_{\underline{w}} p(\underline{y} | \underline{X}, \underline{w}) p(\underline{w}) = \arg\max_{\underline{w}} \left[\log p(\underline{y} | \underline{X}, \underline{w}) + \log p(\underline{w})\right]$$

For $p(\underline{w}) = N(\underline{w} | 0, \lambda^{-1} \underline{\underline{I}})$, $\underline{w}^{MAP} = \arg\max_{\underline{w}} \left[-\frac{1}{2\sigma^2} (\underline{y} - \underline{\underline{X}} \underline{w})^T (\underline{y} - \underline{\underline{X}} \underline{w}) - \frac{\lambda}{2} \underline{w}^T \underline{w}\right] = \left(\underline{\underline{X}}^T \underline{\underline{X}} + \sigma^2 \lambda \underline{\underline{I}}\right)^{-1} \underline{\underline{X}}^T \underline{y}$

Classification

Classification

Classification

- As in regression, we are given a training set $D = \{(\underline{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$ consisting of i/ps $\underline{x}^{(n)} = [x_1^{(n)} \dots x_D^{(n)}]^T$ and o/ps $y^{(n)}$, but the o/ps are now discrete: $y^{(n)} \in \{1, 2 \dots C\}$ where C is the no. of classes / categories
- In classification, we aim to identify a partition of the i/p space into C decision regions, one for each class. Each new i/p is assigned the class of its corresponding decision region, along w/ a measure of confidence.
- The decision regions are separated by decision boundaries — these are pts at which two classes have equal predictive probability.

Problems with methods for regression

- A way of using methods for regression for classification is to let the o/p $\underline{y}^{(n)}$ be a C-dimensional vector w/ one-hot encoding of the class for $\underline{x}^{(n)}$ (if the class is c, $y_i^{(n)} = \mathbb{1}\{i=c\}$).
- We can then solve C linear regression problems, one for each class,

$$\underline{\underline{W}} = (\underline{\underline{\tilde{X}}}^T \underline{\underline{\tilde{X}}})^{-1} \underline{\underline{\tilde{X}}}^T \underline{\underline{Y}}$$

where $\underline{\underline{Y}} = \begin{bmatrix} \leftarrow \underline{y}_1^T \rightarrow \\ \vdots \\ \leftarrow \underline{y}_N^T \rightarrow \end{bmatrix}$ and we predict the class w the highest entry of $\underline{\tilde{x}}^T \underline{\underline{W}}$.

- For the binary case, we could set a simple threshold to determine the class (positive class label if the o/p of the regression model is above the threshold and vice versa).

→ These methods have issues of underrepresenting certain classes or varying the threshold (underfitting hypothesis) as we collect more data pts.

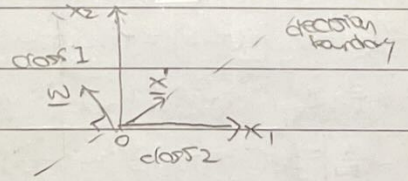
Deterministic linear classification

- Deterministic linear classification works by mapping the o/p of the linear model into discrete class labels. For the binary case, assume $y^{(n)} \in \{0, 1\}$, then we can define

$$\boxed{y^{(n)} = H(\underline{w}^T \underline{\tilde{x}}^{(n)})}$$

where $H(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{o/w} \end{cases}$ is the Heaviside step function.

- The decision boundary is where $\underline{w}^T \underline{x}^{(n)} = 0 \rightarrow \underline{w}$ orthogonal to $\underline{x}^{(n)}$



- This model gives deterministic (hard) predictions — misclassification errors are not allowed and inference is hard (cannot use gradient-based methods to find the MLE)

↑ step function is not differentiable

Probabilistic linear classification

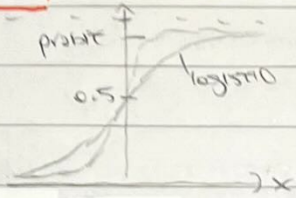
- Probabilistic linear classification works by mapping the o/p of the linear model into class probabilities (between 0 and 1). For the binary case, we define

$$p(y^{(n)}=1 \mid x, \underline{w}) = f(\underline{w}^T \underline{x})$$

where $f(x)$ is a monotonically increasing function that maps $\mathbb{R} \mapsto [0,1]$.

- Typical functions $f(x)$ include:

↳ Logistic function: $\sigma(x) = \frac{1}{1+e^{-x}}$ ↳ Probit function: $\Phi(x) = \int_{-\infty}^{x} \mathcal{N}(z|0,1)\,dx$



- Note that the functions are symmetric about $(0, 0.5)$, so $f(-x) = 1 - f(x)$

- The likelihood function now allows for misclassification errors w/ some probability, and has a non-zero gradient that can be used to find the MLE.

Logistic regression.

- In logistic regression, we use $f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$.

(i) $y^{(n)} \in \{-1, 1\}$

$$p(y^{(n)} \mid x^{(n)}, w) = \frac{1}{1+e^{-y^{(n)} w^T x^{(n)}}} \quad \text{if } y^{(n)}=1 \qquad \frac{e^{-w^T x^{(n)}}}{1+e^{-w^T x^{(n)}}} \quad \text{if } y^{(n)}=0$$

$$\Rightarrow p(y^{(n)} \mid x^{(n)}, w) = \frac{1}{1+e^{-y^{(n)} w^T x^{(n)}}}$$

$$\ln p(\underline{t} \mid \underline{x}, w) = \sum_{n=1}^{N} \ln p(t^{(n)} \mid x^{(n)}, w) = \sum_{n=1}^{N} \ln \sigma(y^{(n)} w^T x^{(n)})$$

Noting that $\frac{\partial \sigma}{\partial a} = \sigma(x)(1-\sigma(x))$,

$$\frac{\partial}{\partial w_j} \ln p(\underline{t} \mid \underline{x}, w) = \sum_{n=1}^{N} \frac{\partial}{\partial w_j} \ln \sigma(y^{(n)} w^T x^{(n)}) = \sum_{n=1}^{N} \frac{1}{\sigma(y^{(n)} w^T x^{(n)})} \sigma(y^{(n)} w^T x^{(n)})(1-\sigma(y^{(n)} w^T x^{(n)})) \, y^{(n)} x_j^{(n)}$$

$$= \sum_{n=1}^{N} (1-\sigma(y^{(n)} w^T x^{(n)})) \, y^{(n)} x_j^{(n)}$$

$$\nabla_w \ln p(\underline{t} \mid \underline{x}, w) = \sum_{n=1}^{N} (1-\sigma(y^{(n)} w^T x^{(n)})) \, y^{(n)} x^{(n)}$$

(ii) $y^{(n)} \in \{0, 1\}$

$$p(y^{(n)} \mid x^{(n)}, w) = \sigma(w^T x^{(n)})^{y^{(n)}} (1-\sigma(w^T x^{(n)}))^{1-y^{(n)}}$$

$$\ln p(\underline{t} \mid \underline{x}, w) = \sum_{n=1}^{N} \ln p(t^{(n)} \mid x^{(n)}, w) = \sum_{n=1}^{N} \left[y^{(n)} \ln \sigma(w^T x^{(n)}) + (1-y^{(n)}) \ln (1-\sigma(w^T x^{(n)})) \right]$$

Noting that $\frac{\partial \sigma}{\partial a} = \sigma(x)(1-\sigma(x))$,

$$\frac{\partial}{\partial w_j} \ln p(\underline{t} \mid \underline{x}, w) = \sum_{n=1}^{N} \left[y^{(n)} \ln \sigma(w^T x^{(n)}) + (1-y^{(n)}) \ln (1-\sigma(w^T x^{(n)})) \right]$$

$$= \sum_{n=1}^{N} \left[\frac{y^{(n)}}{\sigma(w^T x^{(n)})} \sigma(w^T x^{(n)})(1-\sigma(w^T x^{(n)})) x_j^{(n)} - \frac{1-y^{(n)}}{1-\sigma(w^T x^{(n)})} \sigma(w^T x^{(n)})(1-\sigma(w^T x^{(n)})) x_j^{(n)} \right]$$

$$= \sum_{n=1}^{N} \left[y^{(n)} (1-\sigma(w^T x^{(n)})) x_j^{(n)} - (1-y^{(n)}) \sigma(w^T x^{(n)}) x_j^{(n)} \right]$$

$$\nabla_w \ln p(\underline{t} \mid \underline{x}, w) = \sum_{n=1}^{N} \left[y^{(n)} - \sigma(w^T x^{(n)}) \right] x^{(n)}$$

- There is no closed form soln for the MLE → use numerical methods (gradient ascent)

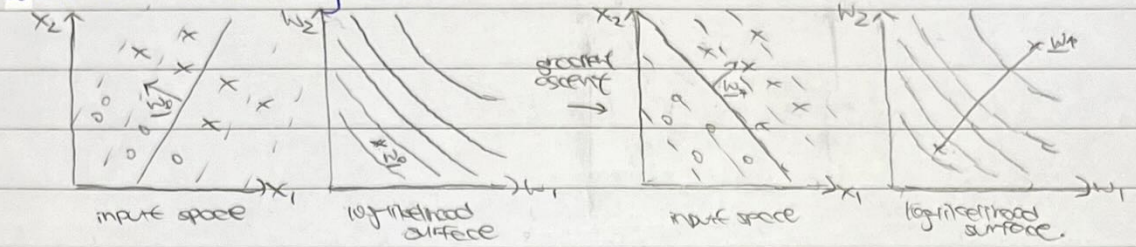
Gradient ascent

- The batch gradient ascent rule to maximise $L(\underline{w})$ is

$$\underline{w}^{new} \leftarrow \underline{w}^{old} + \alpha \frac{dL(\underline{w}^{old})}{d\underline{w}}$$

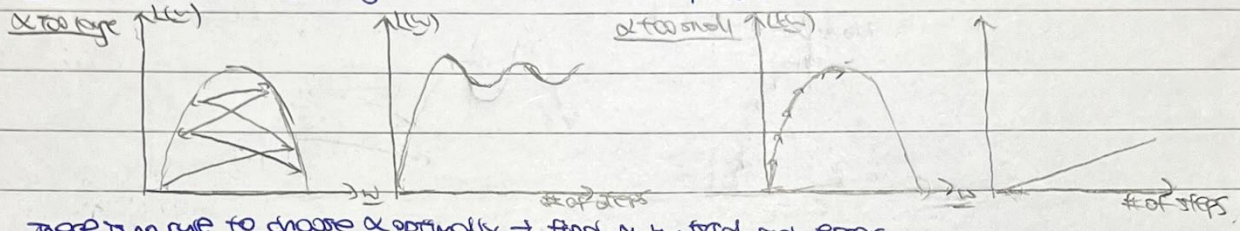
↳ depends on $y^{(n)} \in \{-1, 1\}$ vs $y^{(n)} \in \{0, 1\}$.

where $\alpha > 0$ is the learning rate.



- If α is too large → optimisation bounces around the maximum, and could even diverge.

If α is too small → convergence to the maximum is very slow



There is no rule to choose α optimally → find α by trial and error.

Linear classification with more than 2 classes.

- We can map multiple dps to discrete class labels using the max. function.

$$y^{(n)} = \underset{k \in \{1 \ldots C\}}{\operatorname{argmax}} \; \underline{w}_k^T \underline{x}$$

but this has similar problems as in the deterministic binary classification case

- We can instead use the softmax function to map the dps into class probabilities

equivalent to $p(y^{(n)} | \underline{x}^{(n)}, \underline{w})$ for logistic regression

$$p(y^{(n)} = k | \underline{w}_1 \ldots \underline{w}_C, \underline{x}^{(n)}) = \frac{\exp(\underline{w}_k^T \underline{x}^{(n)})}{\sum_{i=1}^{C} \exp(\underline{w}_i^T \underline{x}^{(n)})}$$

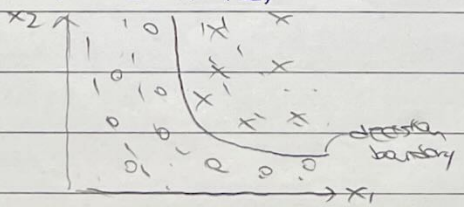
Note this is equivalent to logistic regression when $C = 2$ (softmax is a generalisation of linear regression)

Non-linear logistic regression.

- We can replace $\underline{x}$ w/ non-linear functions of the inputs $\phi(\underline{x}) = [\phi_1(\underline{x}) \ldots \phi_M(\underline{x})]^T$ to convert from linear to non-linear logistic regression.

- Non-linear logistic regression gives non-linear decision boundaries.

* Inference does not change, we just replace $\underline{x}^{(n)}$ w/ $\phi(\underline{x}^{(n)})$

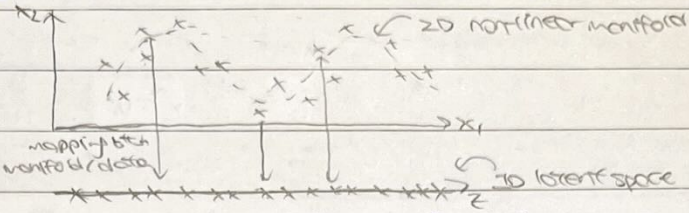


Dimensionality reduction and PCA

Dimensionality reduction and PCA

Dimensionality reduction

- Dimensionality reduction is a type of problem in machine learning requiring to find
  - ↳ A low dimensional representation (manifold) of data
  - ↳ A mapping from data to manifold coordinates and back.
- In general, the resulting manifold is highly nonlinear, but it can be unfolded and mapped non-linearly into a latent plane
- eg: A 1D manifold in 2D space is unfolded into a latent real line

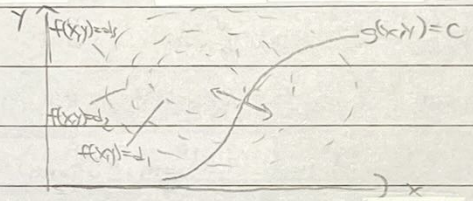


Interpolation btwn pts stays in the data manifold in the 1D latent space, but not in the 2D space.

Lagrangian multipliers.

- The method of Lagrangian multipliers allows us to solve optimisation problems w/ equality constraints

i.e. Maximise $f(x,y)$ subject to $g(x,y) = c$ ← contour line of $g(x,y)$



At the soln, the gradients of $f(x,y)$ and $g(x,y)$ are aligned, $\nabla f(x,y) \parallel \nabla g(x,y)$

↳ The soln should satisfy (i) $g(x,y) = c$, (ii) $\nabla f(x,y) = -\lambda \nabla g(x,y)$

- This is equivalent to maximising the Lagrangian function $L(x,y,\lambda)$ wrt x, y, λ.

$$L(x,y,\lambda) = f(x,y) - \lambda(g(x,y) - c)$$

where λ is the Lagrange multiplier

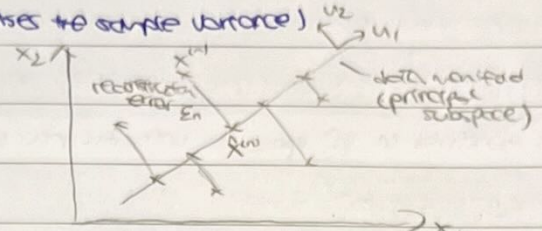
* Note that $\nabla_{x,y} L(x,y,\lambda) = 0 \Leftrightarrow \nabla_{xy} f(x,y) = -\lambda \nabla_{x,y} g(x,y)$; $\nabla_\lambda L(x,y,\lambda) = 0 \Leftrightarrow g(x,y) = c$

- If there are multiple restrictions $h_i(x) = 0$, $i = 1, 2 \ldots$, the Lagrangian becomes $L(x,y,\lambda)$

$$L(x,y,\lambda) = f(x,y) - \sum_i \lambda_i h_i(x,y)$$

Principal component analysis (PCA)

- PCA is a linear dimensionality method — data manifold is assumed to be linear.
- PCA finds the projection that minimises the square reconstruction error (equivalently, the projection that maximises the sample variance)



we can always shift by a const. s.t. this is the case

- Assume we are given a dataset $D = \{x^{(n)}\}_{n=1}^N$ w/ $x^{(n)} \in \mathbb{R}^D$, and $\sum_{n=1}^N x^{(n)} = 0$

Using the orthonormal basis $\{u_i\}_{i=1}^D$, we can rewrite each datapt $x^{(n)}$ as

$$x^{(n)} = \tilde{x}^{(n)} + \xi_n = \sum_{i=1}^M x^{(n)T} u_i u_i + \sum_{i=M+1}^D x^{(n)T} u_i u_i$$

u_i: principal comp. vectors
$x^{(n)T} u_i$: principal comp. score

projected vector $\tilde{x}^{(n)}$ — reconstruction error ξ_n

the latent coordinate in new basis is $z^{(n)} = U x^{(n)}$, where $U = \begin{bmatrix} u_1^T \\ \vdots \\ u_M^T \end{bmatrix}$ is the $M \times D$ projection matrix

- The average squared reconstruction error is given by. (Note $\xi_n = \sum_{i=M+1}^D x^{(n)T} u_i u_i$)

$$\text{cost}(\{u_i\}_{i=1}^D) = \frac{1}{N} \sum_{n=1}^N \xi_n^T \xi_n = \frac{1}{N} \sum_{n=1}^N \left(\sum_{i=M+1}^D x^{(n)T} u_i u_i^T \right) \left(\sum_{j=M+1}^D x^{(n)T} u_j u_j \right)$$

$$= \frac{1}{N} \sum_{n=1}^N \sum_{i=M+1}^D \sum_{j=M+1}^D u_i^T x^{(n)} x^{(n)T} u_j u_i^T u_j = \frac{1}{N} \sum_{n=1}^N \sum_{i=M+1}^D u_i^T x^{(n)} x^{(n)T} u_i$$

orthonormal property: $u_i^T u_j = \begin{cases} 1, & i=j \\ 0, & i \neq j \end{cases}$

$$= \sum_{i=M+1}^D u_i^T \left(\frac{1}{N} \sum_{n=1}^N x^{(n)} x^{(n)T} \right) u_i = \sum_{i=M+1}^D u_i^T \hat{\Sigma} u_i$$

where $\hat{\Sigma} = \frac{1}{N} \sum_{n=1}^N x^{(n)} x^{(n)T}$ is the covariance matrix for the data.

- We use the method of Lagrange multipliers to min. the cost to ensure that $\{u_i\}_{i=1}^D$ has unit norm.

$$L(\{u_i\}_{i=M+1}^D, \{\lambda_j\}_{j=M+1}^D) = \sum_{i=M+1}^D \left[u_i^T \hat{\Sigma} u_i + \lambda_i (1 - u_i^T u_i) \right]$$

cost function; equality constraint

$$\nabla_{u_i} L(\{u_i\}_{i=M+1}^D, \{\lambda_j\}_{j=M+1}^D) = 2 \hat{\Sigma} u_i - 2\lambda_i u_i = 0 \quad \rightarrow \quad \hat{\Sigma} u_i = \lambda_i u_i$$

Note $\frac{\partial}{\partial x}(x^T I x) = \frac{\partial}{\partial x}(x^T I x) = 2x$

$$\nabla_{\lambda_i} L(\{u_i\}_{i=M+1}^D, \{\lambda_j\}_{j=M+1}^D) = 1 - u_i^T u_i = 0 \quad \rightarrow \quad u_i^T u_i = 1$$

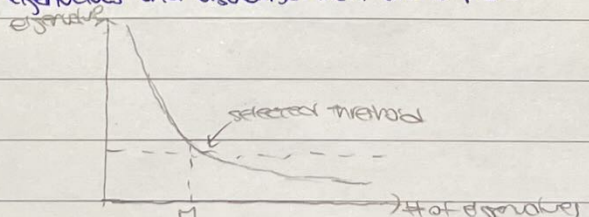
→ (λ_i, u_i) are the eigenvalue and eigenvectors of the covariance matrix $\hat{\Sigma}$.

- We can write the objective function in terms of this soln

$$\text{cost}(\{u_i\}_{i=1}^D) = \sum_{i=M+1}^D u_i^T \hat{\Sigma} u_i = \sum_{i=M+1}^D u_i^T (\lambda_i u_i) = \sum_{i=M+1}^D \lambda_i (u_i^T u_i) = \sum_{i=M+1}^D \lambda_i$$

The PCA soln is given by the $\{u_i\}_{i=1}^D$ s.t. $u_{M+1} \ldots u_D$ are eigenvectors of $\hat{\Sigma}$ w/ the $D-M$ smallest eigenvalues; $u_1 \ldots u_M$ are eigenvectors of $\hat{\Sigma}$ w/ the M largest eigenvalues.

- To choose M, we sort the eigenvalues and visualise them via a plot



We choose M to be the value beyond which there is an inflection pt. The remaining eigenvalues are small w/ eigenvectors that only capture noise.

clustering and EM algorithm

K-means clustering

clustering

- In general, two pts belonging to the same cluster are more similar/closer to each other than two pts belonging to different cluster.
- clustering is an unsupervised learning problem; no labels or rewards.
- clustering maps the dataset $D = \{\underline{x}^{(n)}\}_{n=1}^N$ to a vector of cluster assignments $\underline{s} = \{s_1, \ldots s_N\}$

K-means clustering

we cannot use MLE to find K, o/w we find K to be equal to N

- In K-means clustering, we need to specify the no. of clusters K
- For inputs $D = \{\underline{x}^{(n)}\}_{n=1}^N$, $\underline{x}^{(n)} \in \mathbb{R}^D$, the K-means algorithm is as follows:
  - ↳ From initial (random) cluster means $\underline{m}_k \in \mathbb{R}^D$ for $k = 1 \ldots K$, iterate the two steps
    - (i) Assignment step: $s_n = \underset{k}{\operatorname{argmin}} \|\underline{x}^{(n)} - \underline{m}_k\|_2$ for $n = 1 \ldots N$ [assign to closest cluster]
    - (ii) Update step: $\underline{m}_k = \text{mean}(\underline{x}^{(n)} : s_n = k)$ for $k = 1 \ldots K$ [update the cluster means]
- K-means clustering can be interpreted as optimisation. Let $s_{n,k} = 1$ if data pt n is assigned to cluster k, and zero o/w. (Note $\sum_{k=1}^K s_{n,k} = 1$, i.e. sum of rows of $\underline{\underline{S}}$ equals 1, where $s_{nk} = (\underline{\underline{S}})_{nk}$)

$$C(\{s_{n,k}\}, \{\underline{m}_k\}) = \sum_{n=1}^N \sum_{k=1}^K s_{n,k} \|\underline{x}^{(n)} - \underline{m}_k\|_2^2$$

← could think as PE of a system of springs connecting $\underline{x}^{(n)}$ to $\underline{m}_k$.

K-means tries to minimise the cost function C wrt $\{s_{n,k}\}$ and $\{\underline{m}_k\}$, subject to $\sum_{k=1}^K s_{n,k} = 1$ and $s_{n,k} \in \{0,1\}$

(i) Assignment step: minimise C wrt $\{s_{n,k}\}$, holding $\{\underline{m}_k\}$ fixed.

As each data pt. is independent, minimising C ↔ minimising $\sum_{k=1}^K s_{n,k} \|\underline{x}^{(n)} - \underline{m}_k\|_2^2$ for each row n.

We minimise $\sum_{k=1}^K s_{n,k} \|\underline{x}^{(n)} - \underline{m}_k\|_2^2$ when $s_{n,k} = 1$ for min. $\|\underline{x}^{(n)} - \underline{m}_k\|_2^2$ and zero o/w → $s_n = \underset{k}{\operatorname{argmin}} \|\underline{x}^{(n)} - \underline{m}_k\|_2^2$

(ii) Update step: minimise C wrt $\{\underline{m}_k\}$, holding $\{s_{n,k}\}$ fixed.

As each cluster is independent, minimising C ↔ minimising $\sum_{n:s_n=k} \|\underline{x}^{(n)} - \underline{m}_k\|_2^2$ for each cluster k

$$\frac{dC}{d\underline{m}_k} = \sum_{n:s_n=k} \left[-2\underline{x}^{(n)} + 2\underline{m}_k\right] = 0 \rightarrow \underline{m}_k = \frac{\sum_{n:s_n=k} \underline{x}^{(n)}}{\sum_{n:s_n=k} 1} = \text{mean}(\underline{x}^{(n)} : s_n = k).$$

- The cost function C is a Lyapunov function, so K-means converges. (except for edge cases)

(Finding the global optimum of the cost function C is a NP-hard problem)

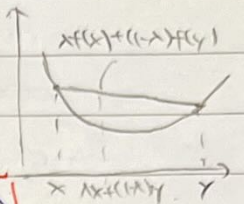
- There are several issues w/ K-means clustering:
  - ↳ The result depends on the initial cluster means $\underline{m}_k$ → use K++ means algorithm to spread out the initial centres, or repeat the algorithm multiple times and pick the best one.
  - ↳ The algorithm returns hard assignments → use EM algorithm for soft assignments (probabilities)
  - ↳ The algorithm assumes spherical clusters → gives strange results when the clusters are not spherical.

EM algorithm

Jensen's inequality

- Jensen's inequality is a general result of convexity. For a convex function f and $\lambda \in [0,1]$

$$\lambda f(x) + (1-\lambda) f(y) \geq f(\lambda x + (1-\lambda) y)$$



we can generalise the result to expectation

$$E[f(x)] \geq f(E[x])$$

* the inequality is reversed for concave functions, so $E[f(x)] \leq f(E[x])$

Kullback-Leibler (KL) divergence

- the KL divergence btwn two probability distributions $p_1(z)$ and $p_2(z)$ is defined as

$$KL(p_1(z) \| p_2(z)) = \sum_z p_1(z) \log \frac{p_1(z)}{p_2(z)}$$

It is a measure of the "distance" / how similar two probability distributions are

- the KL divergence has the following properties:

(i) non-symmetric: $KL(p_1(z) \| p_2(z)) \neq KL(p_2(z) \| p_1(z))$ → "divergence" not "distance"

(ii) Gibbs inequality: $KL(p_1(z) \| p_2(z)) \geq 0$, w/ equality iff $p_1(z) = p_2(z)$. ← proof in JF7 notes

Negative variational free energy $F(\theta, q(\underline{s}))$

- The negative variational free energy / evidence lower bound (ELBO) is a lower bound for the likelihood $\log p(\underline{x}|\theta)$, and is defined to be

$$F(\theta, q(\underline{s})) = \log p(\underline{x}|\theta) - KL(q(\underline{s}) \| p(\underline{s}|\underline{x},\theta)) = \log p(\underline{x}|\theta) - \sum_{\underline{s}} q(\underline{s}) \log \frac{q(\underline{s})}{p(\underline{s}|\underline{x},\theta)}$$

- Note that since KL-divergence is non-negative, we have $F(\theta, q(\underline{s})) \leq \log p(\underline{x}|\theta)$, w/ equality iff $q(\underline{s}) = p(\underline{s}|\underline{x},\theta)$.

- To derive the negative variational free energy $F(\theta, q(\underline{s}))$, consider the log-likelihood $\log p(\underline{x}|\theta)$

$$\log p(\underline{x}|\theta) = \log \left[\sum_{\underline{s}} p(\underline{x},\underline{s}|\theta) \right] = \log \left[\sum_{\underline{s}} p(\underline{s}|\theta)\, p(\underline{x}|\underline{s},\theta) \right] = \log \left[\sum_{\underline{s}} \frac{p(\underline{s}|\theta) p(\underline{x}|\underline{s},\theta)}{q(\underline{s})} q(\underline{s}) \right]$$

$$= \log \left[E_q \left[\frac{p(\underline{s}|\theta) p(\underline{x}|\underline{s},\theta)}{q(\underline{s})} \right] \right] \overset{\text{Jensen}}{\geq} E_q \left[\log \frac{p(\underline{s}|\theta) p(\underline{x}|\underline{s},\theta)}{q(\underline{s})} \right] = F(\theta, q(\underline{s}))$$

$$\therefore F(\theta, q(\underline{s})) = E_q[\log p(\underline{s}|\theta) p(\underline{x}|\underline{s},\theta)] - E_q[\log q(\underline{s})] = \sum_{\underline{s}} q(\underline{s}) \log p(\underline{s}|\theta) p(\underline{x}|\underline{s},\theta) - \sum_{\underline{s}} q(\underline{s}) \log q(\underline{s})$$

this form is equivalent to the form above. Noting that $p(\underline{x},\underline{s}|\theta) = p(\underline{x}|\theta) p(\underline{s}|\underline{x},\theta) = p(\underline{s}|\theta) p(\underline{x}|\underline{s},\theta)$,

$$F(\theta, q(\underline{s})) = E_q[\log p(\underline{x}|\theta) p(\underline{s}|\underline{x},\theta)] - E_q[\log q(\underline{s})] = \sum_{\underline{s}} q(\underline{s}) \log p(\underline{x}|\theta) p(\underline{s}|\underline{x},\theta) - \sum_{\underline{s}} q(\underline{s}) \log q(\underline{s})$$

$$= \log p(\underline{x}|\theta) \sum_{\underline{s}} q(\underline{s}) - \sum_{\underline{s}} q(\underline{s}) \log \frac{q(\underline{s})}{p(\underline{s}|\underline{x},\theta)} = \log p(\underline{x}|\theta) - KL(q(\underline{s}) \| p(\underline{s}|\underline{x},\theta))$$

* $F(\theta, q(\underline{s})) = \log p(\underline{x}|\theta) - KL(q(\underline{s}) \| p(\underline{s}|\underline{x},\theta))$ is useful for the E-step (one term const. w.r.t $q(\underline{s})$)

$F(\theta, q(\underline{s})) = \sum_{\underline{s}} q(\underline{s}) \log p(\underline{s}|\theta) p(\underline{x}|\underline{s},\theta) - \sum_{\underline{s}} q(\underline{s}) \log q(\underline{s})$ is useful for the M-step (one term const. w.r.t θ)

The expectation maximisation (EM) algorithm

- The EM algorithm is an iterative method to find (local) MLE of parameters θ in statistical models which involves unobserved latent variables s.
- Since the negative variational free energy $F(\theta, q(\underline{s}))$ is a lower bound of the log likelihood $p(\underline{x}|\theta)$, optimising $F(\theta, q(\underline{s}))$ gives a better and better lower bound for $\log p(\underline{x}|\theta)$.
- The EM algorithm is as follows:
  - From initial (random) parameters θ_0, iterate $t=1 \dots T$ the two steps
    - (i) E-step: For fixed θ_{t-1}, maximise lower bound $F(\theta_{t-1}, q(\underline{s}))$ wrt $q(\underline{s})$

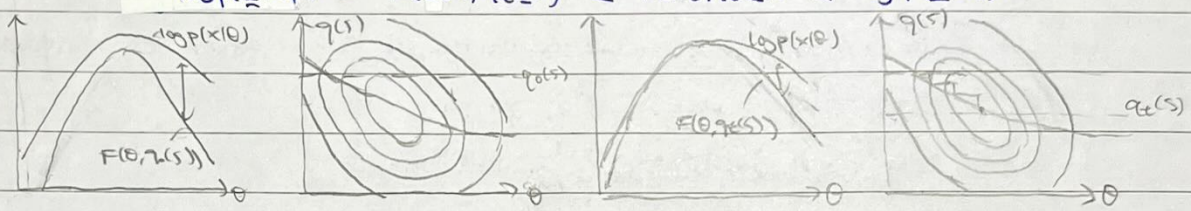
$$F(\theta_{t-1}, q(\underline{s})) = \log p(\underline{x}|\theta_{t-1}) - KL(q(\underline{s}) \| p(\underline{s}|\underline{x}, \theta_{t-1})) \rightarrow \boxed{q(\underline{s}) = p(\underline{s}|\underline{x}, \theta_{t-1})}$$

    - (ii) M-step: For fixed $q_t(\underline{s})$, maximise the lower bound $F(\theta, q_t(\underline{s}))$ wrt θ.

$$F(\theta, q_t(\underline{s})) = \sum_{\underline{s}} q_t(\underline{s}) \log p(\underline{x}, \underline{s}|\theta) p(\underline{s}|\theta) - \sum_{\underline{s}} q_t(\underline{s}) \log q_t(\underline{s}) \rightarrow \boxed{\theta_t = \arg\max_{\theta} \sum_{\underline{s}} q_t(\underline{s}) \log p(\underline{x}, \underline{s}|\theta) p(\underline{s}|\theta)}$$

- Note that each iteration cannot decrease the log likelihood as:

$$\log p(\underline{x}|\theta_{t-1}) \overset{\text{E step}}{=} F(\theta_{t-1}, q_t(\underline{s})) \overset{\text{M step}}{\leq} F(\theta_t, q_t(\underline{s})) \overset{\text{lower bound}}{\leq} \log p(\underline{x}|\theta_t)$$



Mixture of Gaussians (MoG) generative model

- In the MoG model, for each data pt $n=1 \dots N$,
  - the sample cluster membership distribution is given by

$$\boxed{p(s_n = k|\theta) = \pi_k}, \quad \text{where } \sum_{k=1}^{K} \pi_k = 1.$$

  - the sample data value distribution given cluster membership is given by

$$\boxed{p(\underline{x}^{(n)}|s_n = k, \theta) = \mathcal{N}(\underline{x}^{(n)}|\underline{\mu}_k, \underline{\underline{\Sigma}}_k)}$$

- We would like to first learn the parameters θ of the MoG model from data using max. likelihood, $\theta^{ML} = \arg\max_{\theta} \log p(\underline{x}|\theta)$, then infer the cluster membership from the observed data $p(s_n = k|\underline{x}^{(n)}, \theta^{ML})$
- The log-likelihood $\log p(\underline{x}|\theta)$ is given by

$$\log p(\underline{x}|\theta) = \log \prod_{n=1}^{N} p(\underline{x}^{(n)}|\theta) = \sum_{n=1}^{N} \log p(\underline{x}^{(n)}|\theta) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} p(\underline{x}^{(n)}, s_n = k|\theta)$$

$$= \sum_{n=1}^{N} \log \sum_{k=1}^{K} p(s_n = k|\theta)\, p(\underline{x}^{(n)}|s_n = k, \theta) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\underline{x}^{(n)}|\underline{\mu}_k, \underline{\underline{\Sigma}}_k)$$

This can be maximised using the EM algorithm or using direct gradient-based optimisation.

- Direct optimisation is generally faster, but the EM algorithm is often simpler to implement, more widely used and can lead to important generalisations.

Application of the EM algorithm to the MoG model.

- Assume the data is 1D ($x \in \mathbb{R}$) for simplicity. Let the Gaussian mixture model parameters be $\theta = \{\mu_k, \sigma_k^2, \pi_k\}_{k=1}^K$ and 1 latent variable per data pt $s_n \in \{1 \dots K\}$ for $n = 1 \dots N$
- Applying the EM algorithm to the simplified MoG model,

(i) E-step: Note that $p(x^{(n)} | s_n = k, \theta) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left[-\frac{1}{2\sigma_k^2}(x^{(n)} - \mu_k)^2\right]$, $p(s_n = k | \theta) = \pi_k$

$$q(s_n = k) = p(s_n = k | x^{(n)}, \theta) \propto p(x^{(n)} | s_n = k, \theta) p(s_n = k | \theta) = \frac{\pi_k}{\sqrt{2\pi\sigma_k^2}} \exp\left[-\frac{1}{2\sigma_k^2}(x^{(n)} - \mu_k)^2\right] = w_{nk}$$

$$\therefore q(s_n = k) = \frac{w_{nk}}{\sum_{k=1}^K w_{nk}} = r_{nk}$$

($r_{nk} \to 1$: high responsibility; $r_{nk} \to 0$: low responsibility. "responsibility" that component k takes for data pt n)

(ii) M-step: $F(\theta, q(\underline{s})) = \sum_{\underline{s}} q(\underline{s}) \log p(\underline{s} | \theta) p(\underline{x} | \underline{s}, \theta) - \sum_{\underline{s}} q(\underline{s}) \log q(\underline{s})$

$$= \sum_{n=1}^N \sum_{k=1}^K q(s_n = k) \left[\log \pi_k - \frac{1}{2} \log \sigma_k^2 - \frac{1}{2\sigma_k^2}(x^{(n)} - \mu_k)^2\right] + \text{const.}$$

$$\therefore \frac{\partial F}{\partial \mu_j} = \sum_{n=1}^N q(s_n = j) \frac{x^{(n)} - \mu_j}{\sigma_j^2} = 0 \quad \to \quad \mu_j = \frac{\sum_{n=1}^N q(s_n = j) x^{(n)}}{\sum_{n=1}^N q(s_n = j)}$$

$$\frac{\partial F}{\partial \sigma_j^2} = \sum_{n=1}^N q(s_n = j) \left[-\frac{1}{2\sigma_j^2} + \frac{(x^{(n)} - \mu_j)^2}{2\sigma_j^4}\right] = 0 \quad \to \quad \sigma_j^2 = \frac{\sum_{n=1}^N q(s_n = j)(x^{(n)} - \mu_j)^2}{\sum_{n=1}^N q(s_n = j)}$$

$$\frac{\partial (F + \lambda(1 - \sum_k \pi_k))}{\partial \pi_j} = \sum_{n=1}^N \frac{q(s_n = j)}{\pi_j} - \lambda = 0 \quad \to \quad \pi_j = \frac{1}{N} \sum_{n=1}^N q(s_n = j)$$

$\sum_{j=1}^K q(s_n = j) = 1$

$\therefore \sum_{n=1}^N \sum_{j=1}^K q(s_n = j) = N$

- In summary, the EM algorithm for the MoG model is as follows:
  - ↳ From initial (random) parameters $\theta = \{\underline{\mu}_k, \underline{\underline{\Sigma}}_k, \pi_k\}_{k=1}^K$, iterate the two steps:
    - (i) E-step: $r_{nk} = p(s_n = k | \underline{x}^{(n)}, \theta)$ for $n = 1 \dots N$
    - (ii) M-step: $\theta = \arg\max_\theta \sum_{n=1}^N \sum_{k=1}^K r_{nk} \log p(\underline{x}^{(n)}, s_n = k | \theta)$
- Applying the EM algorithm to the MoG model is soft K-means clustering w/ non-axis aligned, non-equally weighted clusters.
- Applying the EM algorithm to the MoG model has several drawbacks:
  - ↳ MoG clusters still have simple shapes → one cluster can have many components / use complex models
  - ↳ ML approach could overfit → use Bayesian approach
  - ↳ coordinate ascent is often slow to converge → joint optimisation of $F(\theta, q(\underline{s}))$ or direct optimisation of the log-likelihood $\log p(\underline{x} | \theta)$
  - ↳ The free energy may diverge (as it is not bounded above) → "Robbon problem" when one of the data pts take the entire cluster.

Sequence modelling

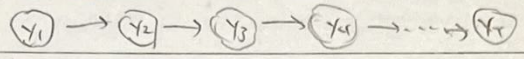
Markov models

Sequence modelling.

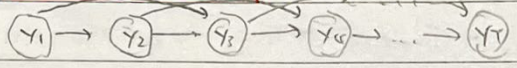
- Sequence modelling can be used to:
  - ↳ Predict future items in a sequence $p(y_{t+1}|y_{1:t})$ [eg: FTSE 100 trends]
  - ↳ Remove noise from a sequence $p(y_{1:t}|y_{1:t})$ [eg: Noise removal]
  - ↳ Predict one sequence from another $p(y'_{1:t}|y_{1:t})$ [eg: translation]
  - ↳ Discover underlying latent variables $p(x_{1:t}|y_{1:t})$ [eg: sentiment analysis]

Markov models.

- A Markov model has the Markov (memoryless) property.
  - (i) 1st order Markov: $y_{t+1} \perp y_{1:t-1} | y_t$ $[p(y_{1:T}) = p(y_1)p(y_2|y_1)p(y_3|y_2)\dots p(y_T|y_{T-1})]$



  - (ii) 2nd order Markov: $y_{t+1} \perp y_{1:t-2} | y_t, y_{t-1}$ $[p(y_{1:T}) = p(y_1)p(y_2|y_1)p(y_3|y_1,y_2)\dots p(y_T|y_{T-2},y_{T-1})]$



- Markov models for discrete data — N-gram models
  - (i) 1st order Markov (bi-gram): $y_t \in \{1\dots K\}$, $p(y_1=k) = \pi_k^0$, $p(y_t=k|y_{t-1}=l) = T_{kl}$
  - (ii) 2nd order Markov (tri-gram): $y_t \in \{1\dots K\}$, $p(y_1=k) = \pi_k^0$, $p(y_t=k|y_{t-1}=l, y_{t-2}=m) = T_{k,l,m}$

- For the bigram model, the marginal distribution for the second state $p(y_2=k)$ is given by

$$p(y_2=k) = \sum_{l=1}^{K} p(y_2=k|y_1=l)\, p(y_1=l) = \sum_{l=1}^{K} T_{kl}\pi_l^0 = \left[\underline{\underline{T}}\,\underline{\pi}^0\right]_k \rightarrow \boxed{p(y_2) = \underline{\underline{T}}\,\underline{\pi}^0}$$

We can find the stationary distribution of the Markov chain by finding the eigenvectors of the transition matrix $\underline{\underline{T}}$ w/ eigenvalue equal to one, since

$$p(y_t=k) = \sum_{l=1}^{K} p(y_t=k|y_{t-1}=l)\,p(y_{t-1}=l) \rightarrow \pi_k^\infty = \sum_{l=1}^{K} T_{kl}\pi_k^\infty \rightarrow \boxed{\underline{\pi}^\infty = \underline{\underline{T}}\,\underline{\pi}^\infty} \quad (\lambda=1)$$

* We can find the max. likelihood transition matrix from an observed sequence by considering the proportions of transitions from state i to state j (out of all transitions from state i)

- Markov models for continuous data — Auto-regressive (AR) models
  - (i) 1st order Markov (AR(1)): $y_t \in \mathbb{R}^D$, $p(y_1) = \mathcal{N}(y_1|\underline{\mu}_0, \underline{\underline{\Sigma}}_0)$, $p(y_t|y_{t-1}) = \mathcal{N}(y_t|\underline{\underline{A}}y_{t-1}, \underline{\underline{\Sigma}})$ (MoG)
    $y_t = \underline{\underline{A}}y_{t-1} + \underline{\underline{\Sigma}}^{1/2}\varepsilon_t$, $\varepsilon_t \sim \mathcal{N}(0, \underline{\underline{I}})$
  - (ii) 2nd order Markov (AR(2)): $y_t \in \mathbb{R}^D$, $p(y_1) = \mathcal{N}(y_1|\underline{\mu}_0, \underline{\underline{\Sigma}}_0)$, $p(y_t|y_{t-1}, y_{t-2}) = \mathcal{N}(y_t|\underline{\underline{A}}_1 y_{t-1} + \underline{\underline{A}}_2 y_{t-2}, \underline{\underline{\Sigma}})$
    $y_t = \underline{\underline{A}}_1 y_{t-1} + \underline{\underline{A}}_2 y_{t-2} + \underline{\underline{\Sigma}}\varepsilon_t$, $\varepsilon_t \sim \mathcal{N}(0, \underline{\underline{I}})$

- For AR(1) and $y \in \mathbb{R}^1$, $p(y_t|y_{t-1}) = \mathcal{N}(y_t|\lambda y_{t-1}, \sigma^2)$ ↳ $y_t = \lambda y_{t-1} + \sigma\varepsilon_t$, $\varepsilon_t \sim \mathcal{N}(0,1)$

Since $p(y_{1:T})$ is Gaussian, and the marginal of a Gaussian is Gaussian, the stationary distribution $p(y_\infty)$ is Gaussian

↳ mean: $E[y_t] = \lambda E[y_{t-1}] + \sigma \cancel{E[\varepsilon_t]}^{0} = \lambda E[y_{t-1}] \rightarrow \mu_t = \lambda\mu_{t-1}$

As $t \to \infty$, $\mu_\infty = \lambda\mu_\infty \rightarrow \boxed{\mu_\infty = 0}$

↳ variance: $E[y_t^2] = \lambda^2 E[y_{t-1}^2] + 2\lambda\sigma \cancel{E[y_{t-1}\varepsilon_t]}^{0} + \sigma^2 \cancel{E[\varepsilon_t^2]}^{1} = \lambda^2 E[y_{t-1}^2] + 2\lambda\sigma \cancel{E[y_{t-1}]}^{0}\cancel{E[\varepsilon_t]}^{0} + \sigma^2 \rightarrow \sigma_t^2 = \lambda^2\sigma_{t-1}^2 + \sigma^2$

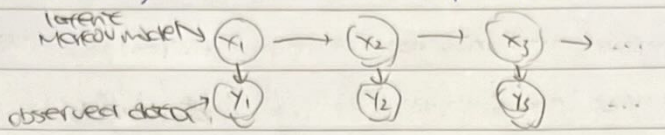
As $t \to \infty$, $\sigma_\infty^2 = \lambda\sigma_\infty^2 + \sigma^2 \rightarrow \boxed{\sigma_\infty^2 = \frac{\sigma^2}{1-\lambda^2}}$

* We always set $\underline{\underline{A}}$ s.t. its eigenvalue has absolute value less than 1 → σ_∞^2 does not blow up

Hidden Markov models

Hidden Markov models

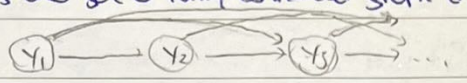
- In hidden Markov models, the real (observed) data y_t depend on latent (unobserved) variables x.



The joint distribution over all the latent variables $x_{1:T}$ and real data $y_{1:T}$ is given by

$$p(y_{1:T}, x_{1:T}) = \prod_{t=1}^{T} p(x_t|x_{t-1}) p(y_t|x_t), \quad \text{where } p(x_1|x_0) = p(x_1)$$

We can marginalise the latents and get a fully-connected graph (not Markovian)



- Two prevalent examples include:
  - ↳ Hidden Markov model (HMM): discrete x, discrete/continuous y
  - ↳ Linear Gaussian state space model (LGSSM): continuous x, continuous y.

Hidden Markov model — discrete hidden state (HMM)

- HMMs have discrete hidden states: $x_t \in \{1 \dots K\}$, $p(x_t = k | x_{t-1} = l) = T_{k,l}$.
- The observed state can be continuous or discrete
  - ↳ continuous observed state: $p(\underline{y}_t | x_t = k) = N(\underline{y}_t | \underline{\mu}_k, \underline{\Sigma}_k)$.
  - ↳ Discrete observed state: $p(y_t = l | x_t = k) = S_{lk}$, where $\underline{S}$ is the emission matrix.
- For discrete hidden state, continuous observed state,

$$p(\underline{y}_1) = \sum_{k=1}^{K} p(\underline{y}_1 | x_1 = k) p(x_1 = k) = \sum_{k=1}^{K} N(\underline{y}_1 | \underline{\mu}_k, \underline{\Sigma}_k) \pi_k^0$$

$$p(\underline{y}_{20}) = \sum_{k=1}^{K} p(\underline{y}_{20} | x_{20} = k) p(x_{20} = k) = \sum_{k=1}^{K} N(\underline{y}_{20} | \underline{\mu}_k, \underline{\Sigma}_k) \pi_k^{20}$$

↖ Mixture of Gaussians

→ we can interpret them as a MoG model w/ dynamic cluster assignments.

Hidden Markov model — continuous hidden state (LGSSM)

- LGSSMs have continuous hidden states: $\underline{x}_t \in \mathbb{R}^K$, $p(\underline{x}_t | \underline{x}_{t-1}) = N(\underline{x}_t | \underline{A}\underline{x}_{t-1}, \underline{Q})$
- The observed state is continuous
  - ↳ continuous observed state: $\underline{y}_t \in \mathbb{R}^D$, $p(\underline{y}_t | \underline{x}_t) = N(\underline{y}_t | \underline{C}\underline{x}_t, \underline{R})$

↖ $D \times K$ matrix

varities of inference

- The four distributional estimates are as follows:

	infer at the same vs sequence? (single state) marginal	(sequence) joint
filter/online	$\boxed{p(x_t \mid y_{1:t})}$	$p(x_{1:t} \mid y_{1:t})$
smoother/offline	$p(x_t \mid y_{1:T})$	$p(x_{1:t} \mid y_{1:T})$

A diff. algorithm is req. for each of the distributional estimates → BFB focuses on $p(x_t|y_{1:t})$

- The two point estimates are as follows:

↳ $\boxed{x_t^* = \underset{x_t}{\operatorname{argmax}}\, p(x_t \mid y_{1:T})}$ ← "bit-optimal"

↳ $\boxed{x'_{1:T} = \underset{x_{1:T}}{\operatorname{argmax}}\, p(x_{1:T} \mid y_{1:T})}$ ← "block-optimal"

- the estimates $x^*_{1:T}$ and $x'_{1:T}$ are the same for the LGSSM (but not discrete hidden state HMM) ↓ shown using counterexample

For LGSSM, $p(x_{1:T} \mid y_{1:T}) = \mathcal{N}(x_{1:T} \mid \mu_{1:T}, \Sigma_{1:T,1:T})$ [Gaussian since $p(x_{1:T}, y_{1:T})$ is Gaussian]

↳ $x^*_t = \underset{x_t}{\operatorname{argmax}}\, p(x_t \mid y_{1:T}) = \mu_t \rightarrow x^*_{1:T} = \mu_{1:T}$

↳ $x'_{1:T} = \underset{x_{1:T}}{\operatorname{argmax}}\, p(x_{1:T} \mid y_{1:T}) = \mu_{1:T}$

Kalman filter

- The kalman filter is an iterative algorithm to compute $p(x_t|y_{1:t})$ for the LGSSM.

(i) start w/ $\boxed{p(x_{t-1} \mid y_{1:t-1}) = \mathcal{N}(x_{t-1} \mid \mu_{t-1}^{t-1}, V_{t-1}^{t-1})}$

(ii) Diffuse via dynamics: use $x_t = A x_{t-1} + Q^{1/2} \varepsilon_t$, $\varepsilon_t \sim \mathcal{N}(0, I)$

$p(x_t \mid y_{1:t-1}) = \int p(x_t \mid x_{t-1}) p(x_{t-1} \mid y_{1:t-1})\, dx_{t-1}$

$\boxed{p(x_t \mid y_{1:t-1}) = \int \mathcal{N}(x_t \mid A x_{t-1}, Q)\, \mathcal{N}(x_{t-1} \mid \mu_{t-1}^{t-1}, V_{t-1}^{t-1})\, dx_{t-1} = \mathcal{N}(x_t \mid \mu_t^{t-1}, V_t^{t-1})}$

where $\boxed{\mu_t^{t-1} = A \mu_{t-1}^{t-1}, \quad V_t^{t-1} = A V_{t-1}^{t-1} A^T + Q}$

(iii) combine w/ likelihood: $p(x_t \mid y_{1:t}) \propto p(x_t \mid y_{1:t-1}) p(y_t \mid x_t)$

$\therefore \boxed{p(x_t \mid y_{1:t}) = \mathcal{N}(x_t \mid \mu_t^t, V_t^t)}$

where $\boxed{\mu_t^t = \mu_t^{t-1} + K_t(y_t - C\mu_t^{t-1}), \quad V_t^t = V_t^{t-1} - K_t C V_t^{t-1}, \quad K_t = V_t^{t-1} C^T (C V_t^{t-1} C^T + R)^{-1}}$

forward algorithm

Markov property allows us to use dynamic programming / belief propagation for efficient computation.

- The forward algorithm is an iterative algorithm to compute $p(x_t|y_{1:t})$ for discrete hidden state HMM.

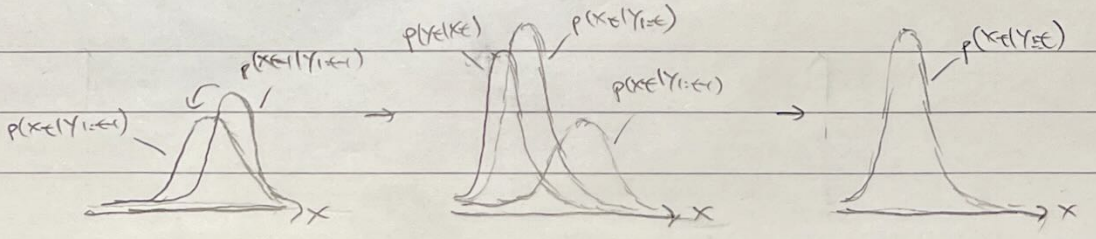
(i) start w/ $\boxed{p(x_{t-1} = k \mid y_{1:t-1}) = p_{t-1}^{t-1}(k)}$

(ii) Diffuse via dynamics: $p(x_t = k \mid y_{1:t-1}) = \sum_{l=1}^{M} p(x_t = k \mid x_{t-1} = l)\, p(x_{t-1} = l \mid y_{1:t-1})$

$\boxed{p_t^{t-1}(k) = \sum_{l=1}^{M} T_{kl}\, p_{t-1}^{t-1}(l)}$

(iii) combine w/ likelihood: $p(x_t = k \mid y_{1:t}) \propto p(x_t = k \mid y_{1:t-1})\, p(y_t \mid x_t = k)$

$\boxed{p_t^t(k) \propto p_t^{t-1}(k)\, p(y_t \mid x_t = k)}$



Derivation of the general filtering equations.

① $p(x_{t+1}|y_{1:t}) = \int p(x_{t+1}|x_t)\, p(x_t|y_{1:t})\, dx_t$ ← replace integrals w/ sums for the discrete case

↳ $p(x_{t+1}|y_{1:t}) = \int p(x_{t+1}, x_t|y_{1:t})\, dx_t = \int p(x_{t+1}|x_t, y_{1:t})\, p(x_t|y_{1:t})\, dx_t$

$= \int p(x_{t+1}|x_t)\, p(x_t|y_{1:t})\, dx_t$ ← cond. indep. from model

② $p(x_{t+1}|y_{1:t}) \propto p(y_t|x_t)\, p(x_t|y_{1:t-1})$ cond. indep. from model

↳ $p(x_t|y_{1:t}) = \frac{p(y_t|x_t, y_{1:t-1})\, p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})} = \frac{p(y_t|x_t)\, p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})}$

$\propto p(y_t|x_t)\, p(x_t|y_{1:t-1})$

Computing the likelihood.

- The likelihood $p(y_{1:T})$ is given by

$$p(y_{1:T}) \prod_{t=1}^{T} p(y_t|y_{1:t-1})$$

where the term $p(y_t|y_{1:t-1})$ is the normalisation const. for computing $p(x_t|y_{1:t})$

→ we simply store $p(y_t|y_{1:t-1})$ for each t and find the product to get the likelihood $p(y_{1:T})$