Unsupervised Learning (in biological data analysis)

Yongjin Park
University of British Columbia
(with Keegan Korthauer, Sara Mostafavi)

24 February, 2022

Little update on the lecture format

- Minimized the lines of codes exposed ...
- However, the `Rmd` file for this lecture is available in the github repo:

`https://github.com/STAT540-UBC/lectures`

What you will learn

General discussions on unsupervised learning

Principal Component Analysis

Clustering to uncover common features and hidden groups

Advanced model-based clustering

Advanced latent variable modelling

Summary

What is Unsupervised Learning?

Supervised learning

- Input: data $\mathcal{X} = \{\mathbf{x}_i, ...\}$
- Input 2: label $\mathcal{Y} = \{y_i, ...\}$
- Goal: learn $f : \mathcal{X} \rightarrow \mathcal{Y}$

X: gene expression samples; Y: disease labels

Unsupervised learning

- Input: data $\mathcal{X} = \{\mathbf{x}_i, ...\}$
- Goal 1: learn $f : \mathcal{Z} \rightarrow \mathcal{X}$
- Goal 2: hidden/latent states, $\mathcal{Z} = \{\mathbf{z}_i, ...\}$

X: gene expression matrices

The goal of unsupervised learning: Find a good representation of high-dimensional data

What is a **good representation** of high-dimensional biology data **to our purpose**?

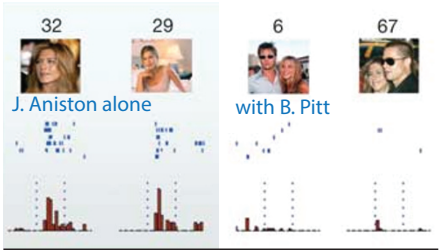
1. What are the properties of high-dimensional data?
   - Continuous vs. discrete? Count vs. intensity?
2. What do we mean by a "good" representation?
3. What is our purpose of unsupervised learning?

Human brain is very good at learning meaningful representations

high-dim. data →  → low-dim. firing pattern

(R. Descartes)

(Digression) Human brain is very good at learning representations



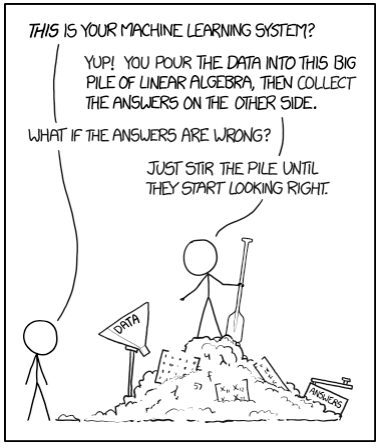
Invariant visual representation by single neurons in the human brain

R. Quian Quiroga[1,2]†, L. Reddy[1], G. Kreiman[3], C. Koch[1] & I. Fried[2,4]

Nature (2005)

- *Warning: just an example of some people's brain*
- *The authors also claimed that they found other very specific neurons (Halle Berry, Sydney Opera House).*

Can you teach a computer to find useful patterns?



From XKCD

Two strategies to learn a "good" representation

Feature engineering

Construct useful combinations of variables, feature engineering

- Principal component analysis (matrix factorization)
- Pattern recognition
- Clustering (average patterns $\approx$ combined features/data)

Manifold learning

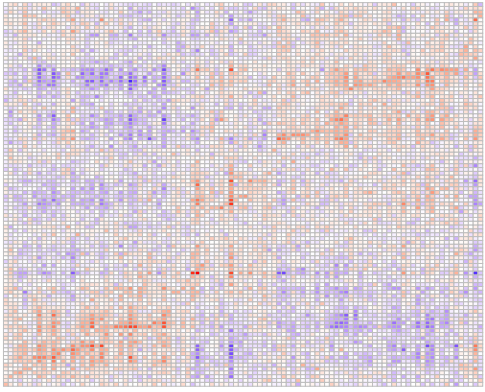
Change the manifold (a coordinate system) of data

- Probabilistic topic models: word frequency $\rightarrow$ simplex
- Angular space $\rightarrow$ Euclidean space
- Observed high-dim. $\rightarrow$ interpretable ones

We use both strategies in practice.

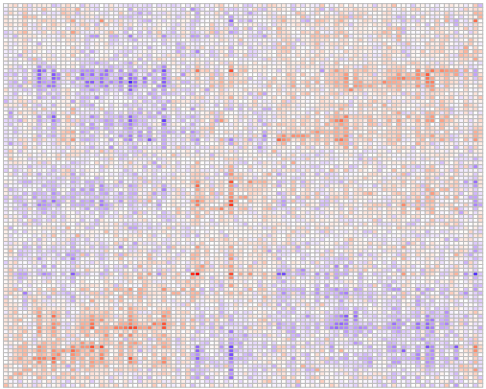
Feature engineering combines multiple vars. to a "factor" var.

$$\mathbf{x}_i = \mathbf{u}_i V^\top + \mathcal{N}(\mathbf{0}, I)$$



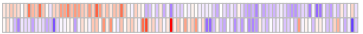
Feature engineering combines multiple vars. to a "factor" var.

$$\mathbf{x}_i = \mathbf{u}_i V^\top + \mathcal{N}(\mathbf{0}, I)$$



features

X

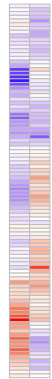


feature loading

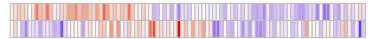
Feature engineering

$$\mathbf{x}_i = \mathbf{u}_i V^\top + \mathcal{N}(\mathbf{0}, I)$$

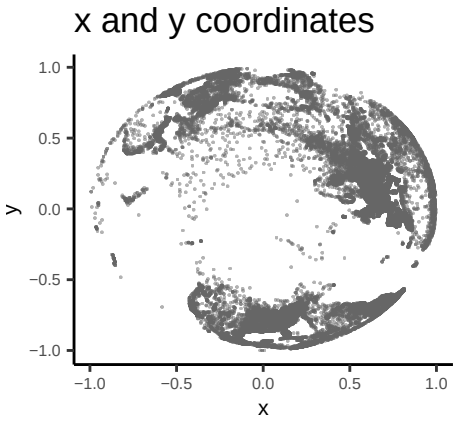
- We call each column of U and V a factor (row/column)
- One factor corresponds to the activities/patterns/expressions of 100 genes/samples
- We will discuss further later.



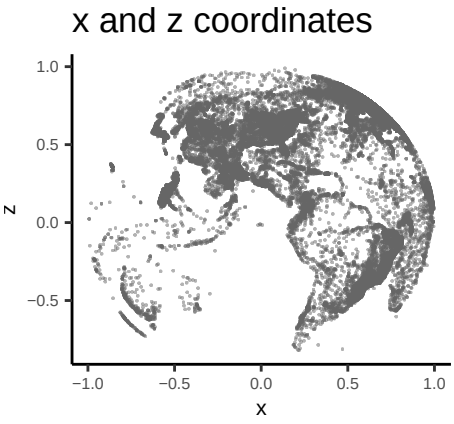
X



Manifold learning transforms the coordinate system of data

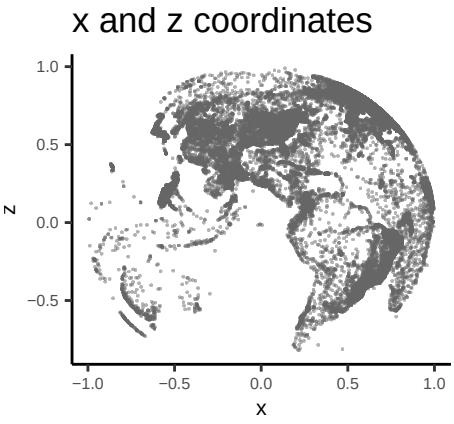


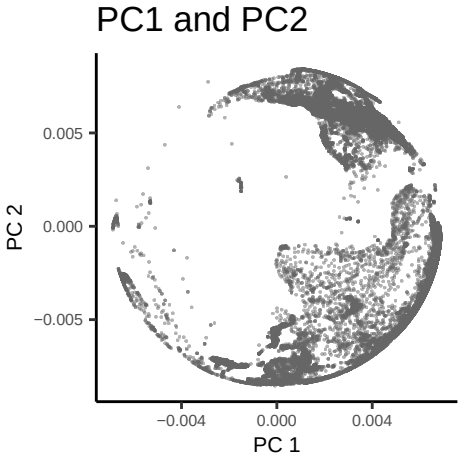
Manifold learning transforms the coordinate system of data



- $x^2 + y^2 + z^2 = 1$
  - Data from https://simplemaps.com/data/world-cities

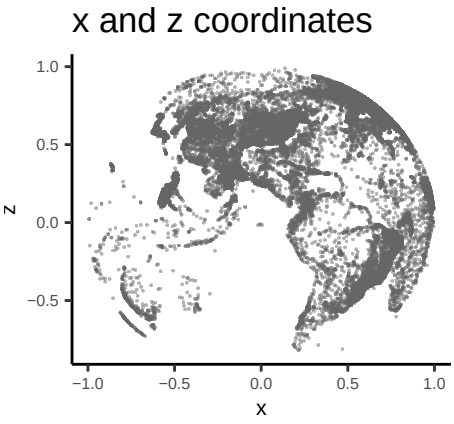
Manifold learning transforms the coordinate system of data

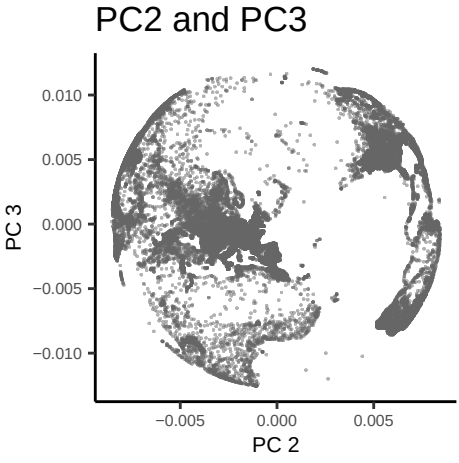




- $x^2 + y^2 + z^2 = 1$
  - Data from https://simplemaps.com/data/world-cities

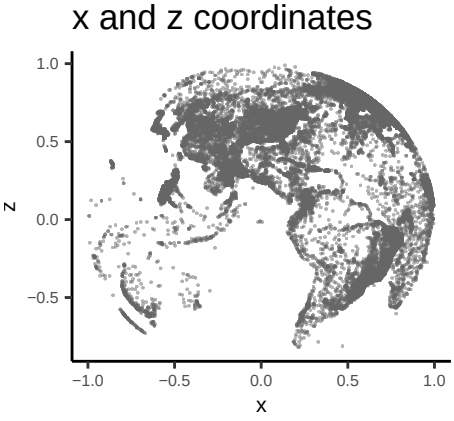
Manifold learning transforms the coordinate system of data

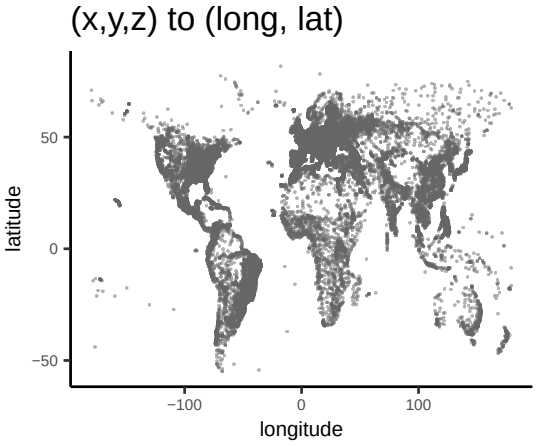




- $x^2 + y^2 + z^2 = 1$
  - Data from https://simplemaps.com/data/world-cities

Manifold learning transforms the coordinate system of data





- $x^2 + y^2 + z^2 = 1$
  - Data from https://simplemaps.com/data/world-cities

Representation learning

Feature engineering

- Combine samples based on similarity to learn common features
- Combine features/genes/proteins to create a new factor
- Matrix factorization to capture factors that explain large variation

Manifold learning

- Change the coordinate system
- What's not separable in one space can be spread out in another one
- Apply non-linear functions to a set of features

What you will learn

General discussions on unsupervised learning

Principal Component Analysis

Clustering to uncover common features and hidden groups

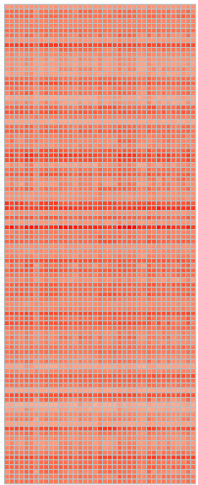
Advanced model-based clustering

Advanced latent variable modelling

Summary

Working example: the photoreceptor data (GSE4051)

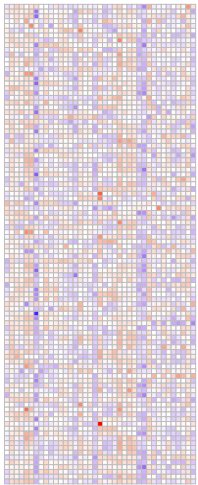
First 100 probes in the data matrix:



- We will call such a high-dimensional matrix X $(m \times n)$
- X has m=45,101 rows (probes/transcripts/genes/features)
- X has n=39 columns (samples/#data points)
- The rows were log-transformed and scaled by `scale()` for visualization
- Each sample is a 45,101-dimensional vector!
- Each gene is a 39-dimensional vector...

Working example: the photoreceptor data (GSE4051)

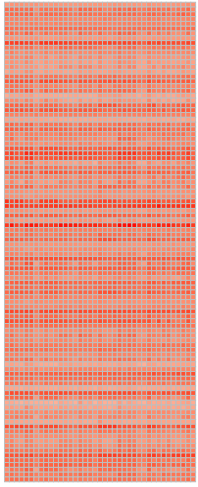
First 100 probes in the data matrix:



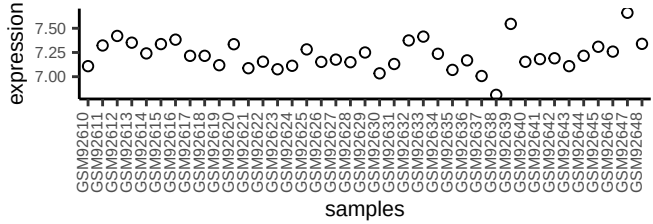
- We will call such a high-dimensional matrix X $(m \times n)$
- X has m=45,101 rows (probes/transcripts/genes/features)
- X has n=39 columns (samples/#data points)
- The rows were log-transformed and scaled by `scale()` for visualization
- Each sample is a 45,101-dimensional vector!
- Each gene is a 39-dimensional vector...

1-dimensional representations

First 100 probes:

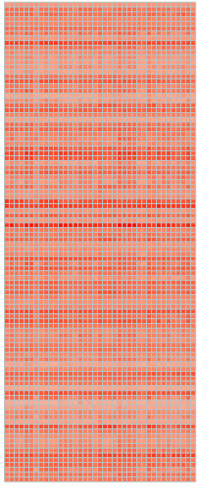


Each probe (row):

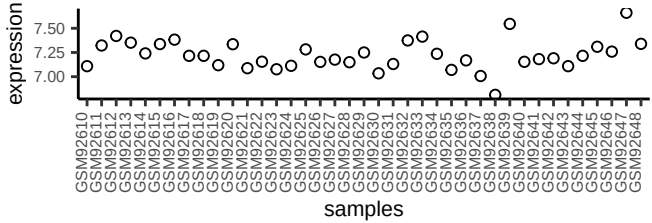


1-dimensional representations

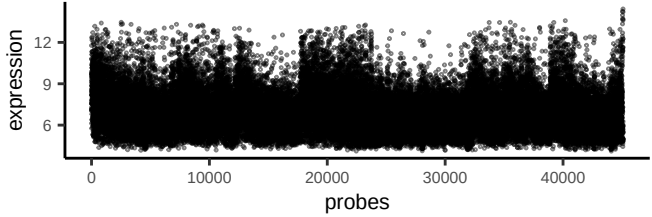
First 100 probes:



Each probe (row):

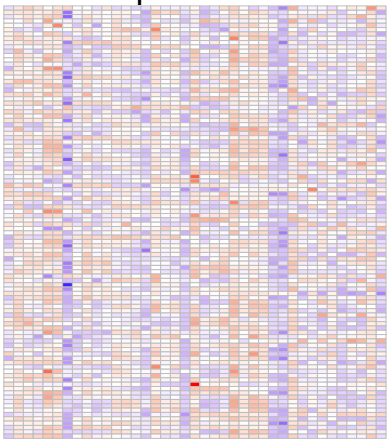


Each sample (column):



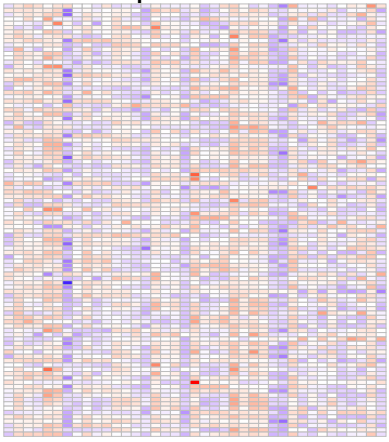
Can you see some common patterns?

first 100 probes

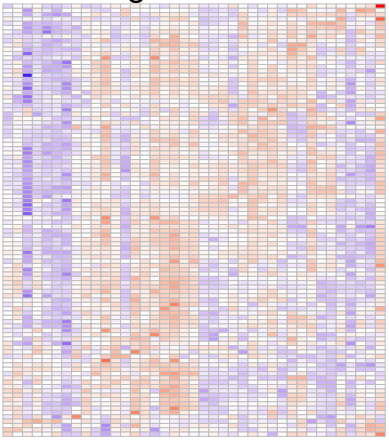


Can you see some common patterns?

first 100 probes



Rearrange them...



PCA: How do we recover "common" patterns from data?

Principal Component Analysis

(Pearson 1901)

- Projection [of the original data] that minimizes the projection cost between the original and projected
- The cost = mean squared distance

(Hotelling 1933)

- Orthogonal projection of data into a lower-dimensional **[principal]** sub-space,
- such that the total **variation of the projected** is maximized

What do we mean by projection to a low-dimensional space?

Recap: We learned that a multivariate linear regression projects a data vector $\mathbf{y}$ onto the column space of the design matrix U:

$$\begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \end{pmatrix} = \begin{pmatrix} U_{11} & \dots & U_{1k} \\ U_{21} & \dots & U_{2k} \\ \vdots & \vdots & \vdots \\ U_{m1} & \dots & U_{mk} \end{pmatrix} \begin{pmatrix} W_1 \\ \vdots \\ W_k \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_m \end{pmatrix}$$

or

$\mathbf{x} = U\mathbf{w} + \epsilon$, for many columns, $X = UW + E$.

- If we **knew** U, we would be able to solve weights W. If $k = 2$, it would result in a projection to the 2D space.
- Unlike regression, we need ask: How do we know this unknown U matrix?

What is a projection matrix?

The purpose of a multivariate regression-based prediction:

$$\mathbf{x} \rightarrow \hat{\mathbf{x}} = U\mathbf{w}$$

The least-square solution for the W:

$$\hat{\mathbf{w}} = (U^\top U)^{-1} U^\top \mathbf{x}$$

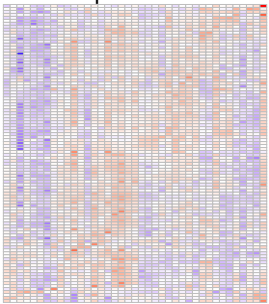
Then we have

$$\hat{\mathbf{x}} = \underbrace{U(U^\top U)^{-1} U^\top}_{\text{projection matrix}} \mathbf{x}$$

You can think of "projection" as **"prediction" in the linear space** defined by the columns of a design matrix.

What will be a good "feature" (the U matrix) to regress on?

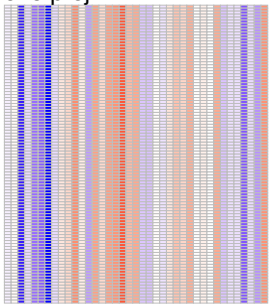
first 100 probes



$5.6287302 \times 10^{-30}$% variance explained

What will be a good "feature" (the U matrix) to regress on?

one proj.

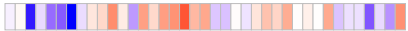


=

1's

X

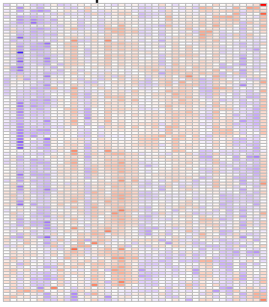
how much contribute?



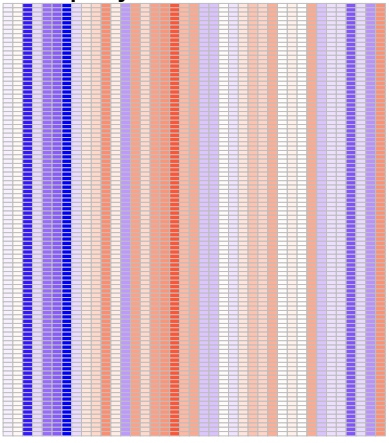
$5.6287302 \times 10^{-30}$% variance explained

What will be a good "feature" (the U matrix) to regress on?

first 100 probes



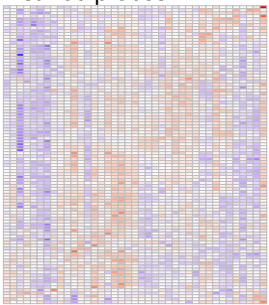
one proj.



$5.6287302 \times 10^{-30}$% variance explained

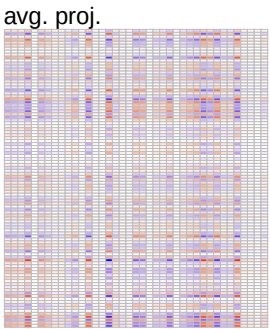
How about an average vector?

first 100 probes



1.4864225% variance explained

How about an average vector?



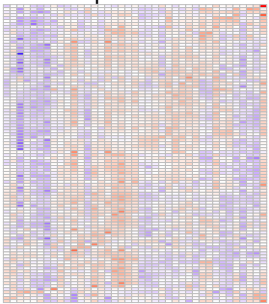
=

x

1.4864225% variance explained

How about an average vector?

first 100 probes

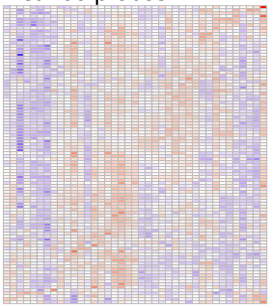


avg. proj.

1.4864225% variance explained

Let's try out random projection

first 100 probes

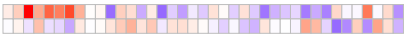


2.5639553% variance explained

Let's try out random projection

=

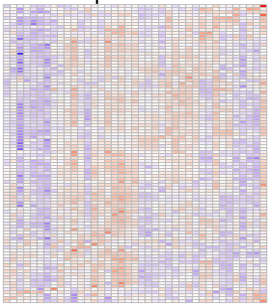
x



2.5639553% variance explained

Let's try out random projection

first 100 probes



random proj.

2.5639553% variance explained

Can we find a set of "good" vectors to maximize the explained variability?

Recap: Sample covariance matrix

- Sample mean: $\bar{X}_i = \sum_{j=1}^{m} X_{ji}/m$
- Sample variance: $\sum_{j=1}^{m}(X_{ji} - \bar{X}_i)^2/(m-1)$
- Sample covariance between i and k:

$$\frac{1}{m-1}\sum_{j=1}^{m}(X_{ji} - \bar{X}_i)(X_{jk} - \bar{X}_k)$$

If all the column vectors $\mathbf{x}_i$ are standardized, the column-by-column covariance $X^\top X/(m-1)$.

If all the row vectors $\mathbf{x}_d$ are standardized, the row-by-row covariance $XX^\top/(n-1)$.

Total variance of the projected data

Total variance

Given the projected, $\hat{X} = \mathbf{u}_1 \cdot (W_{11}, \ldots, W_{1n})$, our goal is

$$\max \mathbb{V}\left[\hat{X}\right]$$

- We have two unknown variables U and W
- There are an infinite number of solutions.

Constrained total variance

Given the projected, $\hat{X} = \mathbf{u}_1 \mathbf{w}$, and a unit vector, namely $\|\mathbf{u}_1\| = 1$, our goal is equivalent to

$$\max \mathbf{u}^\top X X^\top \mathbf{u}$$

Because each $\hat{W}_i$ is the solution to the least-square problem:

$$\hat{W}_i = \arg\min \|\mathbf{x}_i - \mathbf{u} W_i\|$$

by solving the least square:

$$\hat{W}_i = \mathbf{x}_i^\top \mathbf{u} / \mathbf{u}^\top \mathbf{u}, \ \forall i$$

Principal Component Analysis as an eigen value problem

PCA

Letting the feature-by-feature sample covariance matrix $\hat{\Sigma} = XX^\top/(n-1)$, we want to find a unit vector $\mathbf{u}$ by

$$\max \mathbf{u}^\top \hat{\Sigma} \mathbf{u}$$

subject to $\mathbf{u}^\top \mathbf{u} = 1$.

Eigen value problem

Given the covariance matrix $\hat{\Sigma}$, we can resolve an eigen-value λ and the corresponding eigen-vector $\mathbf{u}$ such that

$$\hat{\Sigma}\mathbf{u} = \lambda \mathbf{u}$$

("eigen" mean "own" in German).

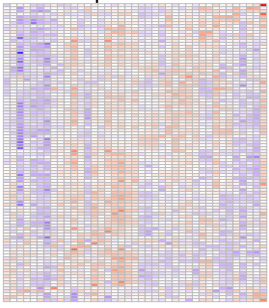
Why are they equivalent? Solving the PCA problem is ...

$$\iff \max \underbrace{\mathbf{u}^\top \hat{\Sigma} \mathbf{u}}_{\text{total variation}} + \underbrace{\lambda\,(1 - \mathbf{u}^\top \mathbf{u})}_{\text{constraint}},\ \lambda > 0 \quad \text{a.k.a. Lagrangian}$$

Taking the derivative with respect to $\mathbf{u}$ and setting it to zero, we get the eigen-value problem.

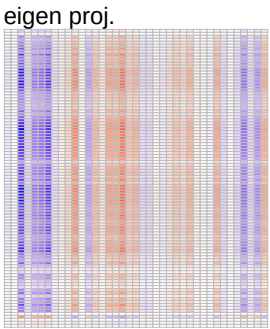
Let's see how much the first eigenvector can explain

first 100 probes

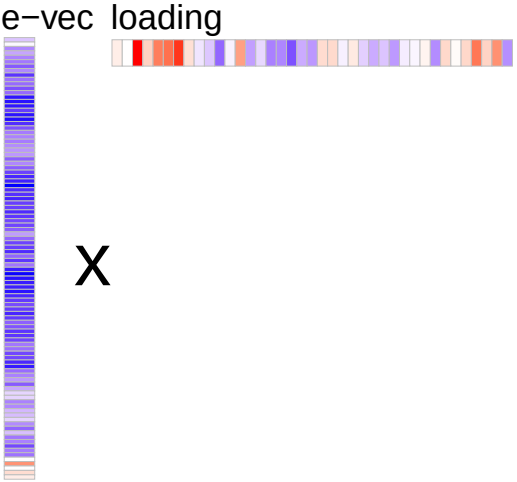


11.7121305% variance explained

Let's see how much the first eigenvector can explain



=

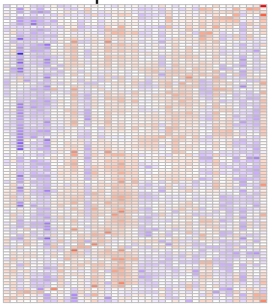


x

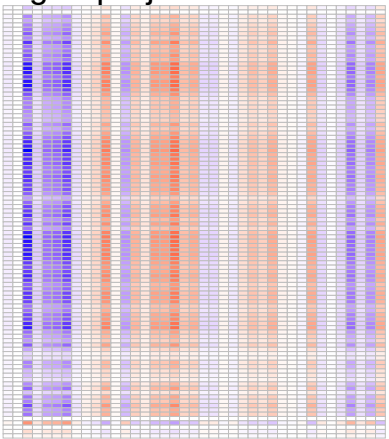
11.7121305% variance explained

Let's see how much the first eigenvector can explain

first 100 probes



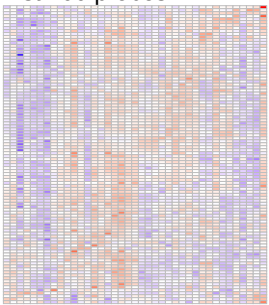
eigen proj.



11.7121305% variance explained

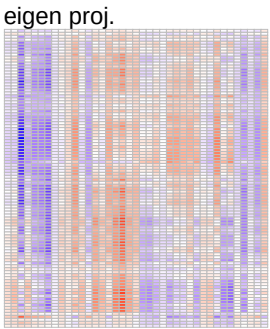
Top two eigen-vectors

first 100 probes

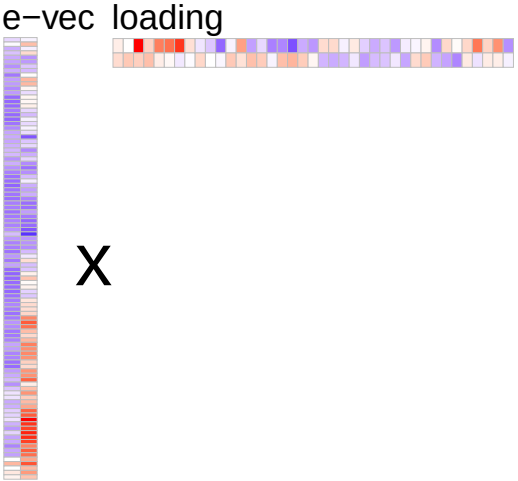


35.3132403% variance explained

Top two eigen-vectors



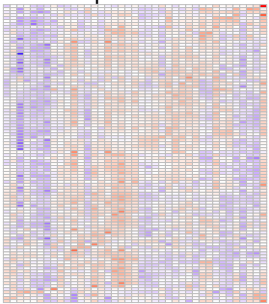
=



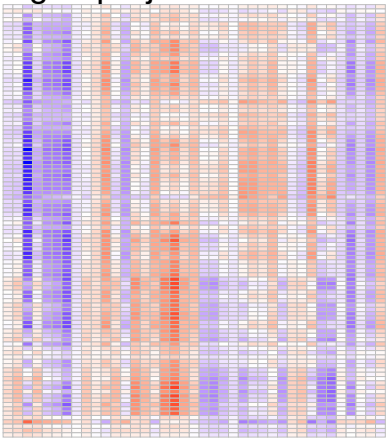
35.3132403% variance explained

Top two eigen-vectors

first 100 probes



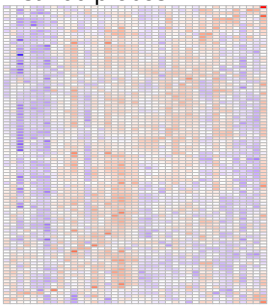
eigen proj.



35.3132403% variance explained

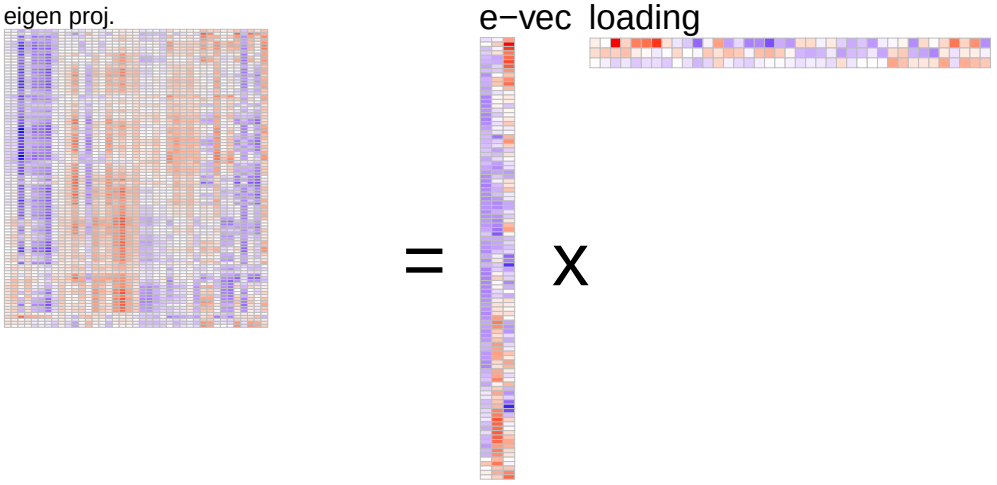
Top three eigen-vectors

first 100 probes



48.6387999% variance explained

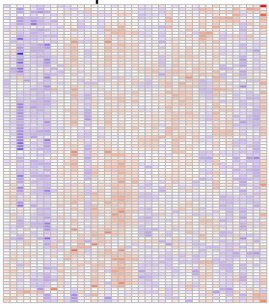
Top three eigen-vectors



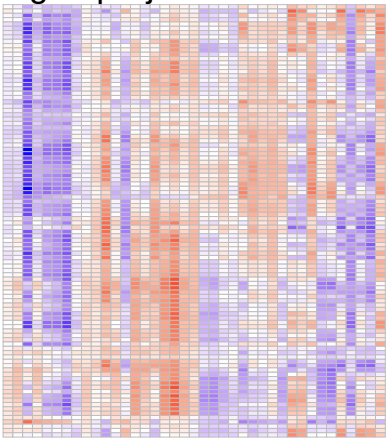
48.6387999% variance explained

Top three eigen-vectors

first 100 probes



eigen proj.



48.6387999% variance explained

SVD: another equivalent method for PCA

Singular Value Decomposition

SVD identifies three matrices of X:

$$X = UDV^\top$$

where both U and V vectors are orthonormal, namely,

- $U^\top U = I$, $\mathbf{u}_k^\top \mathbf{u}_k = 1$ for all k,
- $V^\top V = I$, $\mathbf{v}_k^\top \mathbf{v}_k = 1$ for all k.

Covariance by SVD

Covariance across the columns (samples)

$$X^\top X/(m-1) = VD^2V^\top/(m-1)$$

Covariance across the rows (genes)

$$XX^\top/(n-1) = UD^2U^\top/(n-1)$$

Remark: standaridzed matrix

SVD: another equivalent method for PCA

We can confirm the equivalent relations by multiplying singular vectors to the covariance matrix:

$$\underbrace{\left(\frac{1}{m-1}X^\top X\right)}_{\text{sample covariance}}\mathbf{v}_1 = \frac{1}{m-1}(\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k)\begin{pmatrix} D_1^2 & 0 & \dots & \dots \\ 0 & D_2^2 & 0 & \dots \\ 0 & \dots & \ddots & 0 \\ 0 & \dots & 0 & D_k^2 \end{pmatrix}\begin{pmatrix} \mathbf{v}_1^\top \\ \mathbf{v}_2^\top \\ \vdots \\ \mathbf{v}_k^\top \end{pmatrix}\mathbf{v}_1$$

SVD: another equivalent method for PCA

We can confirm the equivalent relations by multiplying singular vectors to the covariance matrix:

$$\underbrace{\left(\frac{1}{m-1}X^\top X\right)}_{\text{sample covariance}}\mathbf{v}_1 = \frac{1}{m-1}(\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k)\begin{pmatrix} D_1^2 & 0 & \dots & \dots \\ 0 & D_2^2 & 0 & \dots \\ 0 & \dots & \ddots & 0 \\ 0 & \dots & 0 & D_k^2 \end{pmatrix}\begin{pmatrix} \mathbf{v}_1^\top \\ \mathbf{v}_2^\top \\ \vdots \\ \mathbf{v}_k^\top \end{pmatrix}\mathbf{v}_1$$

$$= \frac{1}{m-1}(\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k)\begin{pmatrix} D_1^2 & 0 & \dots & \dots \\ 0 & D_2^2 & 0 & \dots \\ 0 & \dots & \ddots & 0 \\ 0 & \dots & 0 & D_k^2 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

SVD: another equivalent method for PCA

We can confirm the equivalent relations by multiplying singular vectors to the covariance matrix:

$$
\begin{aligned}
\underbrace{\left(\frac{1}{m-1}X^\top X\right)}_{\text{sample covariance}}\mathbf{v}_1 &= \frac{1}{m-1}(\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k)\begin{pmatrix} D_1^2 & 0 & \dots & \dots \\ 0 & D_2^2 & 0 & \dots \\ 0 & \dots & \ddots & 0 \\ 0 & \dots & 0 & D_k^2 \end{pmatrix}\begin{pmatrix} \mathbf{v}_1^\top \\ \mathbf{v}_2^\top \\ \vdots \\ \mathbf{v}_k^\top \end{pmatrix}\mathbf{v}_1 \\
&= \frac{1}{m-1}(\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k)\begin{pmatrix} D_1^2 & 0 & \dots & \dots \\ 0 & D_2^2 & 0 & \dots \\ 0 & \dots & \ddots & 0 \\ 0 & \dots & 0 & D_k^2 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \\
&= \frac{1}{m-1}(\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k)\begin{pmatrix} D_1^2 \\ 0 \\ 0 \\ 0 \end{pmatrix}
\end{aligned}
$$

SVD: another equivalent method for PCA

We can confirm the equivalent relations by multiplying singular vectors to the covariance matrix:

$$\underbrace{\left(\frac{1}{m-1}X^\top X\right)}_{\text{sample covariance}}\mathbf{v}_1 = \frac{1}{m-1}(\mathbf{v}_1,\mathbf{v}_2,\dots,\mathbf{v}_k)\begin{pmatrix} D_1^2 & 0 & \dots & \dots \\ 0 & D_2^2 & 0 & \dots \\ 0 & \dots & \ddots & 0 \\ 0 & \dots & 0 & D_k^2 \end{pmatrix}\begin{pmatrix} \mathbf{v}_1^\top \\ \mathbf{v}_2^\top \\ \vdots \\ \mathbf{v}_k^\top \end{pmatrix}\mathbf{v}_1$$

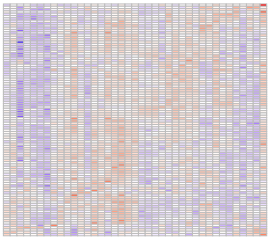
$$= \frac{1}{m-1}(\mathbf{v}_1,\mathbf{v}_2,\dots,\mathbf{v}_k)\begin{pmatrix} D_1^2 & 0 & \dots & \dots \\ 0 & D_2^2 & 0 & \dots \\ 0 & \dots & \ddots & 0 \\ 0 & \dots & 0 & D_k^2 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

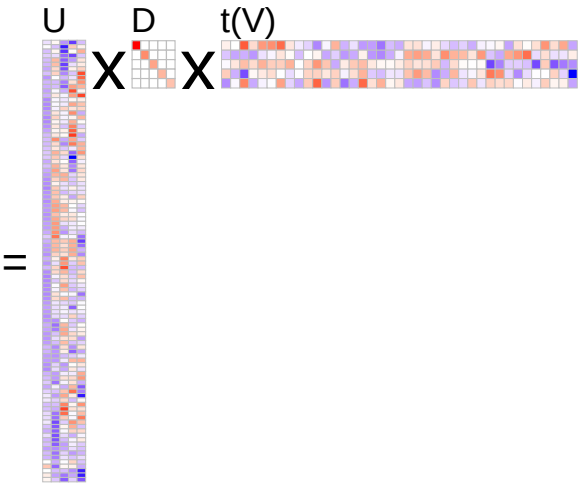
$$= \frac{1}{m-1}(\mathbf{v}_1,\mathbf{v}_2,\dots,\mathbf{v}_k)\begin{pmatrix} D_1^2 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \underbrace{\frac{D_1^2}{m-1}}_{\text{eigenvalue}}\underbrace{\mathbf{v}_1}_{\text{eigenvector}}$$

Run SVD to find many principal components

```
svd.out <- svd(x.sub, nu = 5, nv = 5)
U <- svd.out$u; D <- diag(svd.out$d[1:5]); V <- svd.out$v
```

data matrix

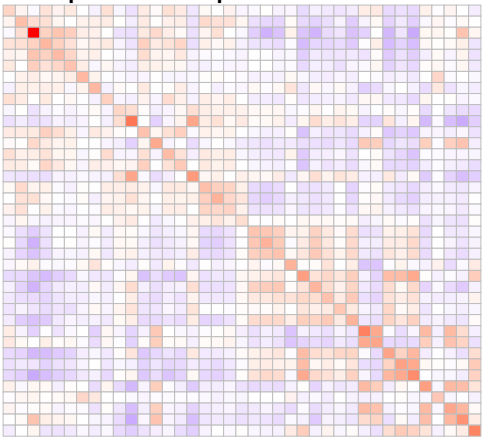




(Co-)Variance decomposition

$$\hat{\Sigma} = X^\top X/(m-1)$$

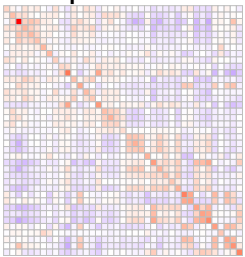
sample x sample covariance



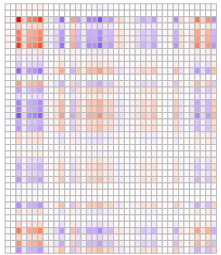
(Co-)Variance decomposition

$$\hat{\Sigma} = \frac{X^\top X}{m-1} = V\frac{D^2}{m-1}V^\top = \sum_{k=1} \lambda_k \mathbf{v}_k \mathbf{v}_k^\top, \quad \lambda_k = \frac{D_k^2}{m-1}$$

sample x sample

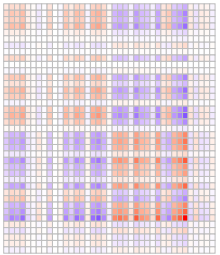


k 1, lambda: 16.9



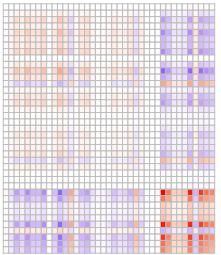
+

k 2, lambda: 5.84



+

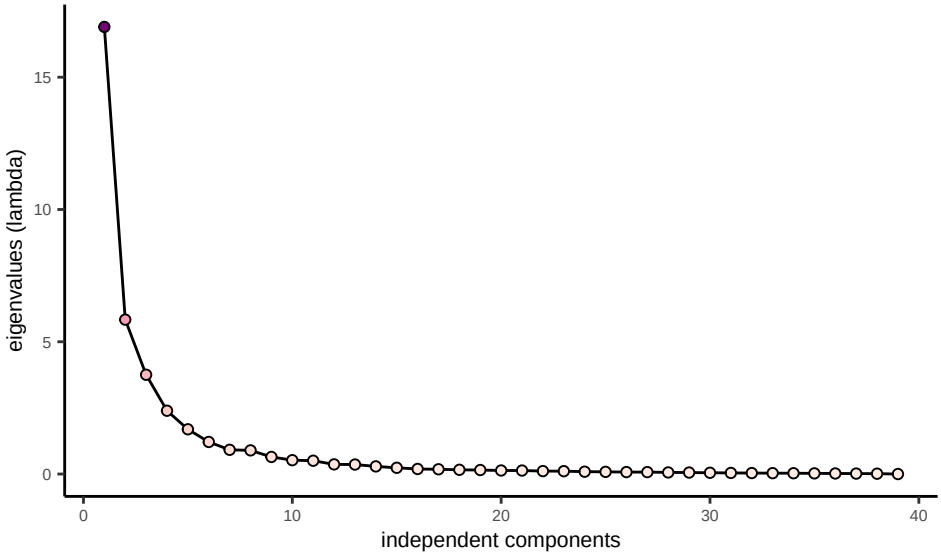
k 3, lambda: 3.75



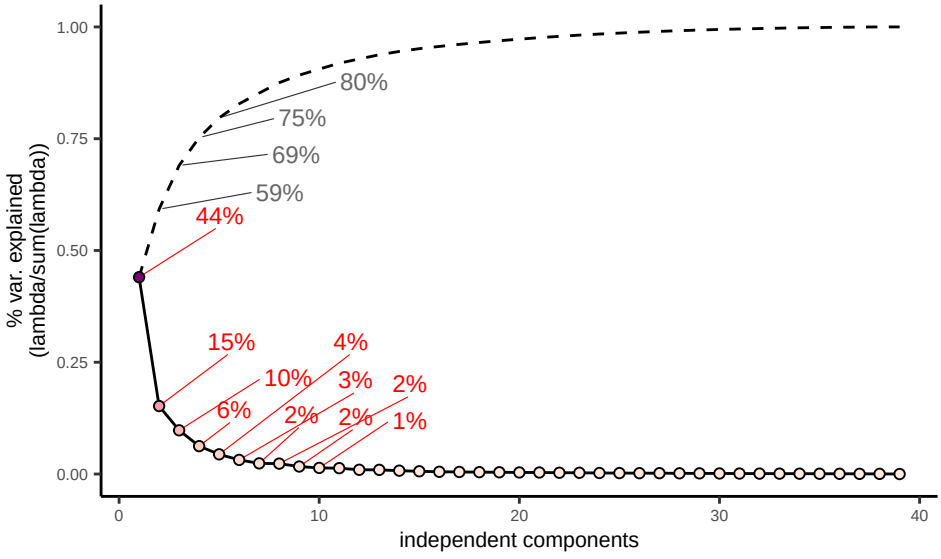
+

- How much variance is explained by each component?

Eigenvectors decompose total covariance



Eigenvectors decompose total covariance



What you will learn

General discussions on unsupervised learning

Principal Component Analysis

Clustering to uncover common features and hidden groups

Advanced model-based clustering

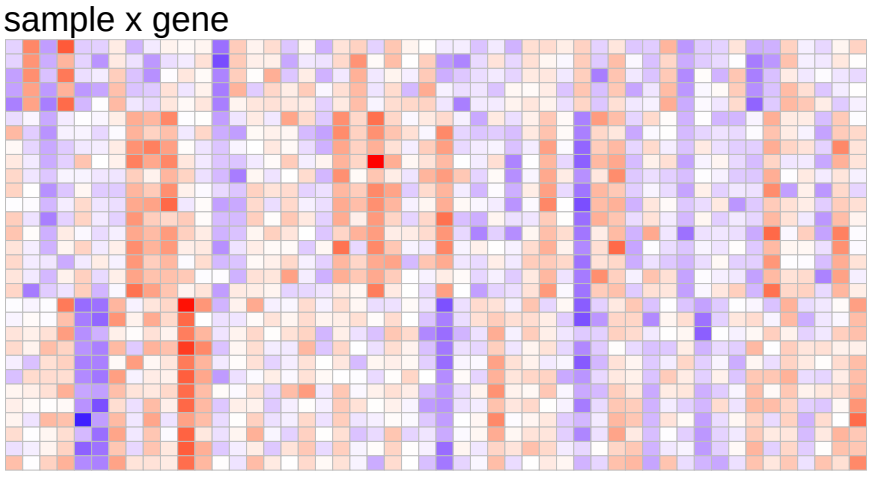
Advanced latent variable modelling

Summary

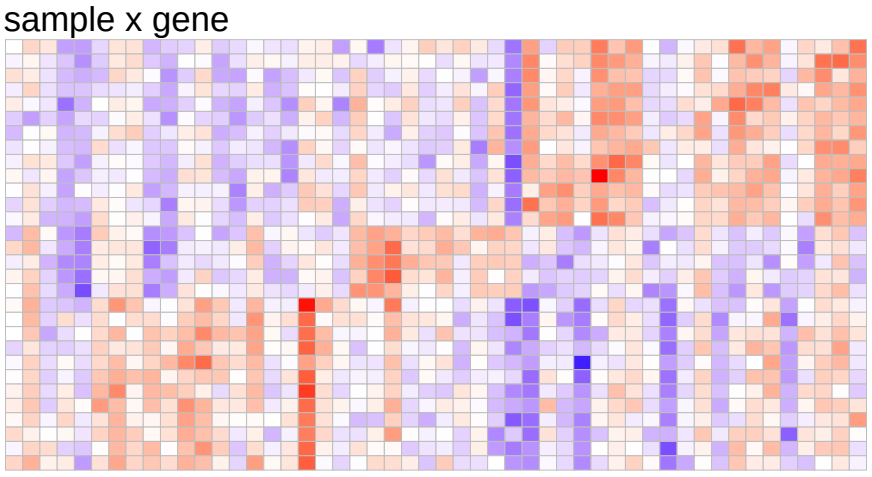
The goal of this clustering section

- Model-based unsupervised learning
  - Latent variable model
- Expectation Maximization
- Hands-on experience with clustering algorithm

What clustering can do: Find common patterns across rows and columns



What clustering can do: Find common patterns across rows and columns



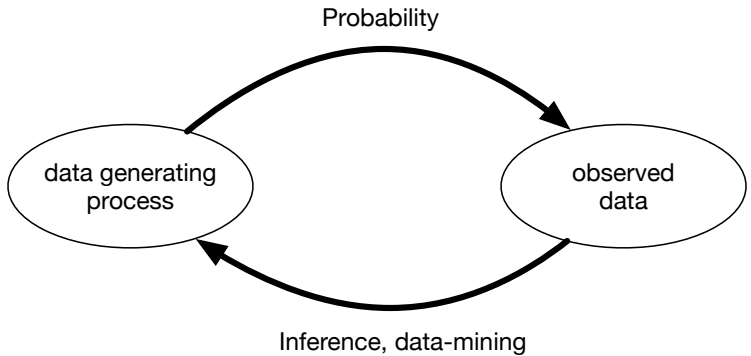
Model-based clustering (if you were a Bayesian statistician...)

Simulation/data generation

Given a data generating process, what are the properties of the outcomes?

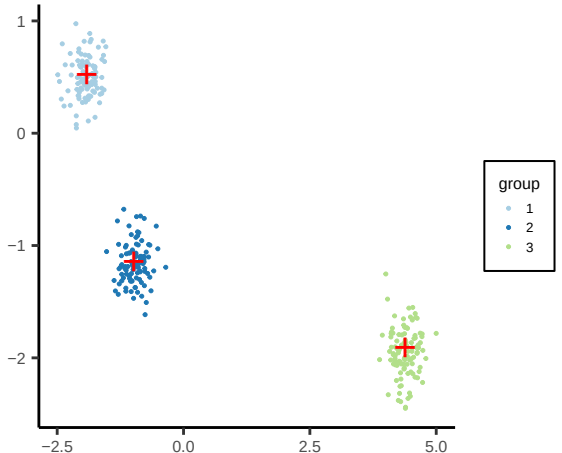
Model inference

Given the outcomes, what can we say about the process that generated the data?



Wassermann, *All of Statistics* (2004)

What do we want to know from data?

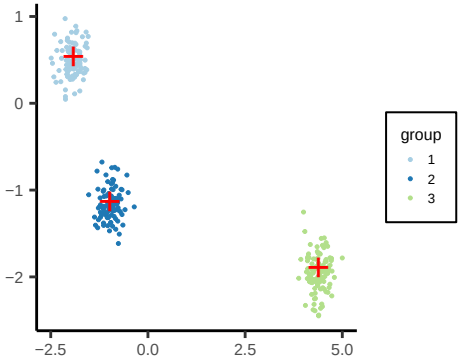


Two goals: recover (1) group membership and (2) the centroids (red marks)

A chicken-and-egg problem: guessing latent membership vs. parametric inference

- If we knew the membership of all the points, we can simply estimate the centre (e.g., taking sample mean within each cluster)
- If we knew the centre coordinates, we would be able to assign points to most probable groups easily based on distance from the centre points.
- Statistical answer: Solve the underlying inference problem.
- To a parameter estimator, the membership assignments are *hidden* (latent).

Let's think about the data-generating process to "reverse" it

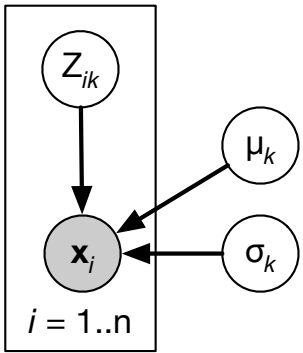


Latent Variable Model

What could have been done to generate observe data?

- Latent membership: Z_{ik}
- Model parameters: μ_k and σ_k

Gaussian Mixture Model (k-means)



How do we infer Z and μ, σ?

GMM data generating process

1. Initialize μ_k (the centre of each group) and σ_k (the spread within each group)
2. Randomly assign group membership, $Z_{ik} = 1$ iff a point i belongs to a group k.
3. Generate: $\mathbf{x}_i | Z_{ik} = 1, \mu_k \sim \mathcal{N}(\mu_k, \sigma^2 I)$

Maximum Likelihood from Incomplete Data via the *EM* Algorithm

By A. P. DEMPSTER, N. M. LAIRD and D. B. RUBIN

Harvard University and Educational Testing Service

[Read before the ROYAL STATISTICAL SOCIETY at a meeting organized by the RESEARCH SECTION on Wednesday, December 8th, 1976, Professor S. D. SILVEY in the Chair]

SUMMARY

A broadly applicable algorithm for computing maximum likelihood estimates from incomplete data is presented at various levels of generality. Theory showing the monotone behaviour of the likelihood and convergence of the algorithm is derived. Many examples are sketched, including missing value situations, applications to grouped, censored or truncated data, finite mixture models, variance component estimation, hyperparameter estimation, iteratively reweighted least squares and factor analysis.

Keywords: MAXIMUM LIKELIHOOD; INCOMPLETE DATA; EM ALGORITHM; POSTERIOR MODE

Expectation Maximization for GMM MLE

$$J \equiv \log \prod_{i=1}^{n} p(\mathbf{x}_i|\mu, \sigma)$$

Expectation Maximization for GMM MLE

$$
\begin{aligned}
J &\equiv \log \prod_{i=1}^{n} p(\mathbf{x}_i|\mu,\sigma) \\
&= \log \prod_{i=1}^{n} \sum_{Z} p(\mathbf{x}_i|Z,\mu,\sigma)p(Z)
\end{aligned}
$$

* It might be difficult to enumerate all the Z's... so let's introduce some other distributions that will help our "guessing" work, which we call it $q(Z)$

EM algorithm: What is the best way to guess latent variables?

$$\sum_i \log p(\mathbf{x}_i|\mu,\sigma) \quad = \quad \sum_{i=1}^{n} \log \sum_{Z_i} \frac{q(Z_i)}{q(Z_i)} p(\mathbf{x}_i|Z_i,\mu,\sigma)p(Z_i)$$

EM algorithm: What is the best way to guess latent variables?

$$\sum_i \log p(\mathbf{x}_i|\mu,\sigma) = \sum_{i=1}^{n} \log \sum_{Z_i} \frac{q(Z_i)}{q(Z_i)} p(\mathbf{x}_i|Z_i,\mu,\sigma)p(Z_i)$$

$$\underset{\text{Jansen}}{\geq} \sum_{i=1}^{n} \sum_{Z_i} q(Z_i) \log \frac{p(\mathbf{x}_i|Z_i,\mu,\sigma)p(Z_i)}{q(Z_i)}$$

EM algorithm: What is the best way to guess latent variables?

$$\sum_i \log p(\mathbf{x}_i|\mu,\sigma) = \sum_{i=1}^{n} \log \sum_{Z_i} \frac{q(Z_i)}{q(Z_i)} p(\mathbf{x}_i|Z_i,\mu,\sigma)p(Z_i)$$

$$\underset{\text{Jansen}}{\geq} \sum_{i=1}^{n} \sum_{Z_i} q(Z_i) \log \frac{p(\mathbf{x}_i|Z_i,\mu,\sigma)p(Z_i)}{q(Z_i)}$$

What is the best $q(Z)$?

EM algorithm: optimal E-step is to take the posterior probability

If $q(Z) = p(Z|\mathbf{x}_i, \mu, \sigma)$ (by Bayes rule),

$$= \sum_{i}^{n} \sum_{Z_i} p(Z_i|\mathbf{x}_i, \mu, \sigma) \frac{p(\mathbf{x}_i|Z_i, \mu, \sigma)p(Z_i)}{p(Z_i|\mathbf{x}_i\mu, \sigma)}$$

EM algorithm: optimal E-step is to take the posterior probability

If $q(Z) = p(Z|\mathbf{x}_i, \mu, \sigma)$ (by Bayes rule),

$$
\begin{aligned}
&= \sum_{i}^{n} \sum_{Z_i} p(Z_i|\mathbf{x}_i, \mu, \sigma) \frac{p(\mathbf{x}_i|Z_i, \mu, \sigma) p(Z_i)}{p(Z_i|\mathbf{x}_i \mu, \sigma)} \\
&= \sum_{i}^{n} \sum_{Z_i} p(Z_i|\mathbf{x}_i, \mu, \sigma) \frac{p(\mathbf{x}_i, Z_i|\mu, \sigma) p(\mathbf{x}_i|\mu, \sigma)}{p(\mathbf{x}_i, Z_i|\mu, \sigma)}
\end{aligned}
$$

EM algorithm: optimal E-step is to take the posterior probability

If $q(Z) = p(Z|\mathbf{x}_i, \mu, \sigma)$ (by Bayes rule),

$$
\begin{aligned}
&= \sum_i^n \sum_{Z_i} p(Z_i|\mathbf{x}_i, \mu, \sigma) \frac{p(\mathbf{x}_i|Z_i, \mu, \sigma)p(Z_i)}{p(Z_i|\mathbf{x}_i\mu, \sigma)} \\
&= \sum_i^n \sum_{Z_i} p(Z_i|\mathbf{x}_i, \mu, \sigma) \frac{p(\mathbf{x}_i, Z_i|\mu, \sigma)p(\mathbf{x}_i|\mu, \sigma)}{p(\mathbf{x}_i, Z_i|\mu, \sigma)} \\
&= \sum_{i=1}^n \underbrace{\left[\sum_{Z_i} p(Z_i|\mathbf{x}_i, \mu, \sigma) \right]}_{=1} \log p(\mathbf{x}_i|\mu, \sigma) \\
&= \sum_i \log p(\mathbf{x}_i|\mu, \sigma)
\end{aligned}
$$

The inequality becomes equality.

Expectation Maximization algorithm = expected MLE

The goal:

$$\log p(X|\mu,\sigma) = \log \sum_Z p(X,Z|\mu,\sigma) \tag{1}$$

$$\geq_{\text{Jensen}} \sum_Z \underbrace{p(Z|X,\mu,\sigma)}_{\text{E-step}} \underbrace{\log p(X|Z,\mu,\sigma)}_{\text{M-step}} \tag{2}$$

$$= \underbrace{\mathbb{E}_{p(Z|X,\mu,\sigma)}}_{\text{E-step}} \left[\underbrace{\log p(X|Z,\mu,\sigma)}_{\text{M-step}} \right] \tag{3}$$

(Z is discrete, e.g., a membership indicator)

Solution: **Maximize** the lower bound by taking **the expectation** over the posterior probability.

EM algorithm of GMM: E-step

Log-likelihood under some group (μ_k and σ_k):

$$\log p(\mathbf{x}_i|\mu_k, \sigma_k) = \log \mathcal{N}(\mathbf{x}_i|\mu_k, \sigma_k)$$

How to estimate the posterior?

$$p(Z_{ik}|\mathbf{x}_i, \mu, \sigma) = \frac{\exp\{\log p(\mathbf{x}_i|\mu_k, \sigma_k)\}}{\sum_{k'} \exp\{\log p(\mathbf{x}_i|\mu_{k'}, \sigma_{k'})\}}$$

Remark: We can stochastically sample $Z_{ik} = 1$ with the posterior probability.

EM algorithm of GMM: M-step

Maximization step to optimize model parameters

Let this expected lower-bound (ELBO)

$$\mathcal{L}(\mathbf{x}_i; \{\mu_k\}, \{\sigma_k\}) = \sum_{i=1}^{n} \sum_{k=1}^{K} Z_{ik} \log p(\mathbf{x}_i | Z_{ik}, \mu_k, \sigma_k)$$

Given Z, what are the unknown? We can take gradient steps (e.g., `torch`)

$$\mu_k^{(t)} \leftarrow \mu_k^{(t-1)} + \rho \nabla_{\mu_k} \sum_i \mathcal{L}(\mathbf{x}_i)$$

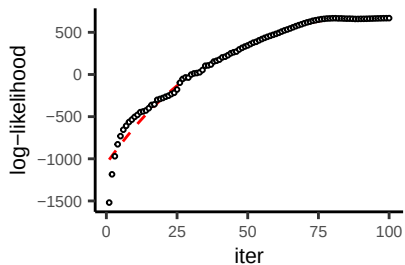
$$\sigma_k^{(t)} \leftarrow \sigma_k^{(t-1)} + \rho \nabla_{\sigma_k} \sum_i \mathcal{L}(\mathbf{x}_i)$$

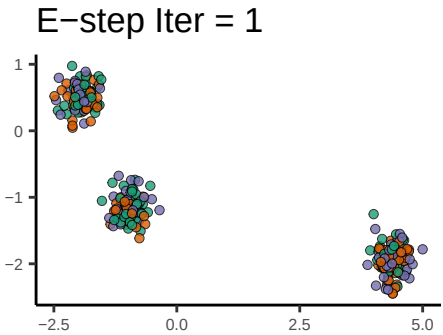
Remark: We have an analytical solution for μ and σ in this example.

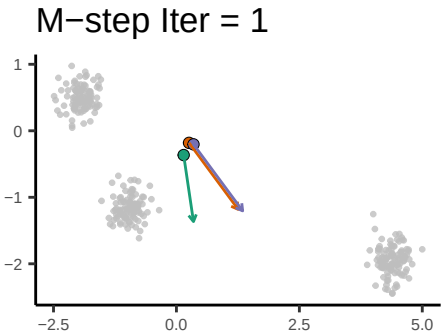
Alternate E- and M-step until convergence

```
for(tt in 1:100){
    rand.idx <- take.estep()
    z <- nnf_one_hot(rand.idx)
    llik <- take.mstep(z)
}
```

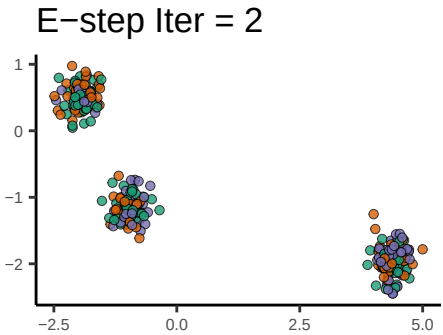
Find the details here:
`https://github.com/STAT540-UBC/lectures`

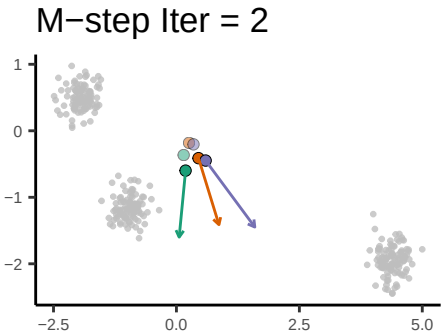




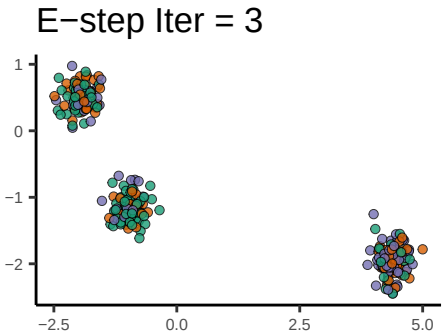


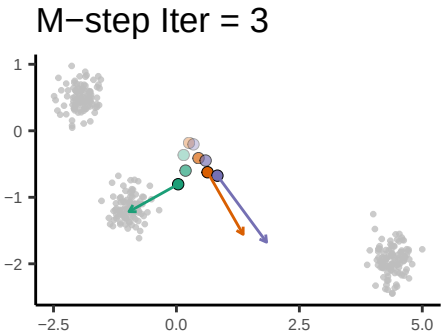
- Arrows: stochastic gradient $\nabla\mu$
- Colour: latent membership



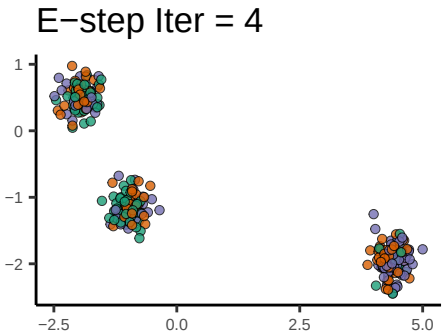


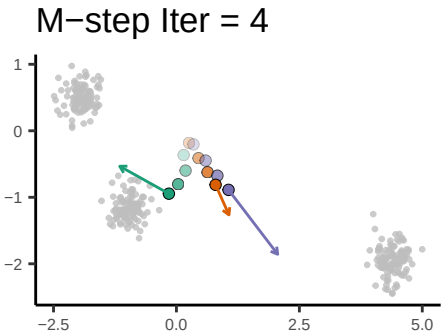
- Arrows: stochastic gradient $\nabla\mu$
- Colour: latent membership



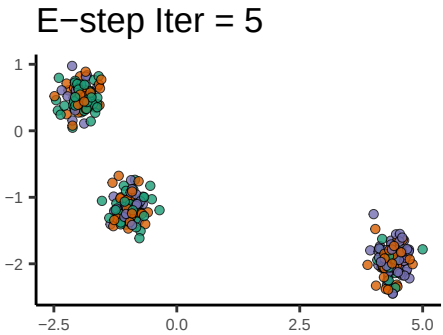


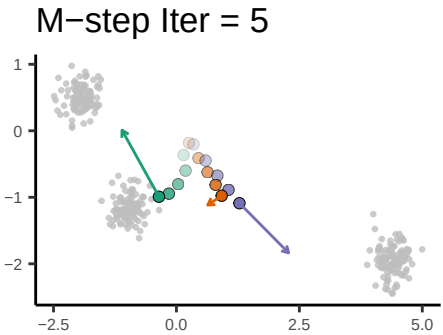
- Arrows: stochastic gradient $\nabla\mu$
- Colour: latent membership



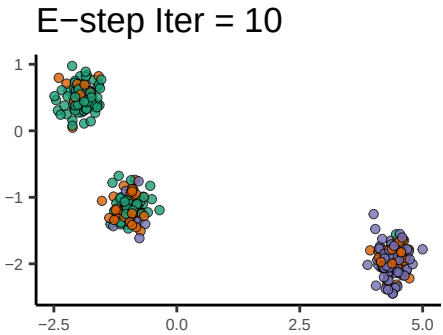


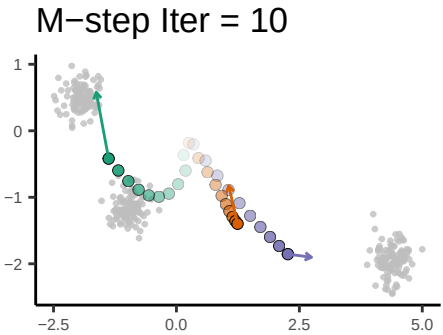
- Arrows: stochastic gradient $\nabla\mu$
- Colour: latent membership



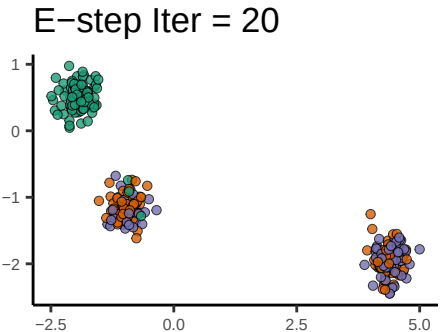


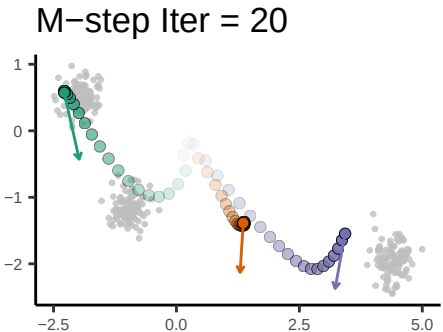
- Arrows: stochastic gradient $\nabla\mu$
- Colour: latent membership



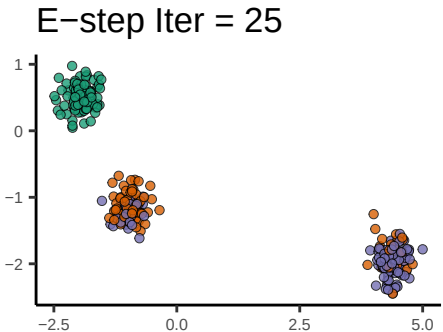


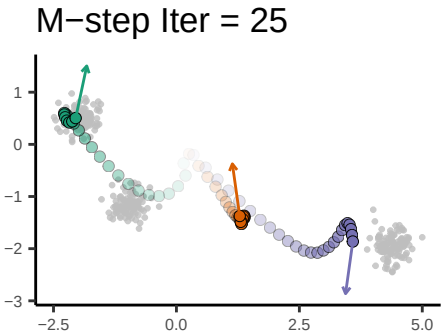
- Arrows: stochastic gradient $\nabla\mu$
- Colour: latent membership



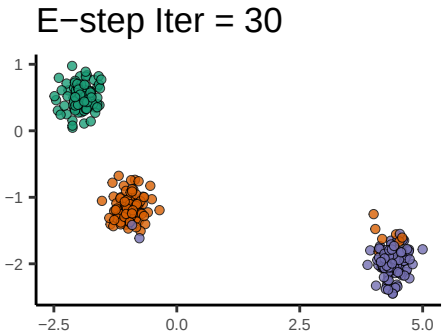


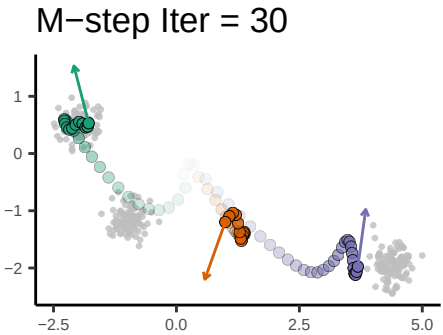
- Arrows: stochastic gradient $\nabla\mu$
- Colour: latent membership



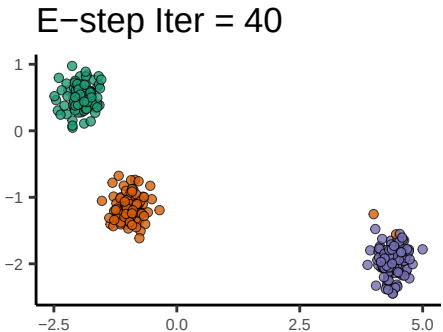


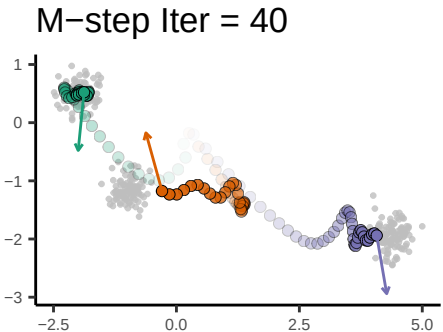
- Arrows: stochastic gradient $\nabla\mu$
- Colour: latent membership



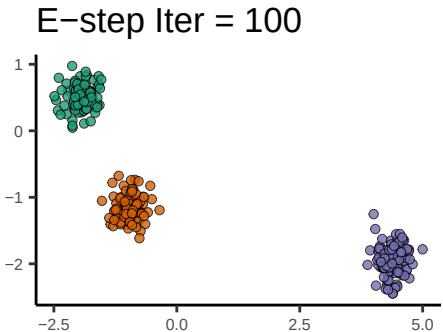


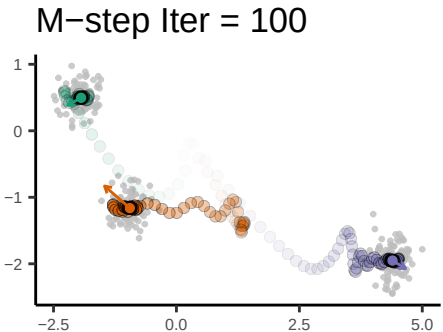
- Arrows: stochastic gradient $\nabla\mu$
- Colour: latent membership





- Arrows: stochastic gradient $\nabla\mu$
- Colour: latent membership





- Arrows: stochastic gradient $\nabla\mu$
- Colour: latent membership

Discussion on (stochastic) k-means clustering algorithm

- What if we initialize to "poor" centre points?
- How do we know the number of clusters?
- Random seeding the latent membership vs. random centre coordinates?
- What does the alternating EM algorithm guarantee?
- Why numerical optimization, rather than setting the gradient zero?

What you will learn

General discussions on unsupervised learning

Principal Component Analysis

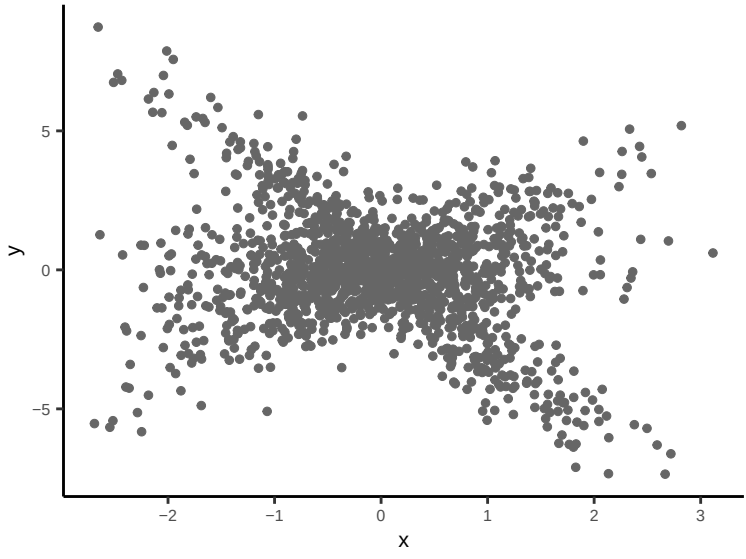
Clustering to uncover common features and hidden groups

Advanced model-based clustering

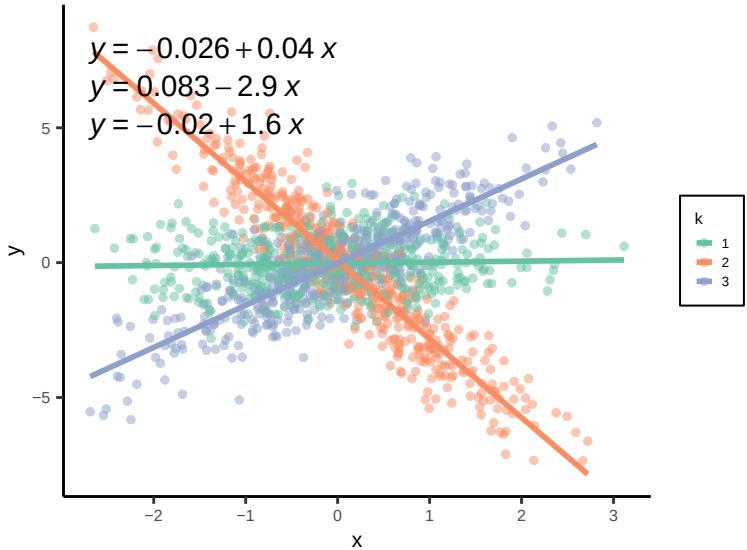
Advanced latent variable modelling

Summary

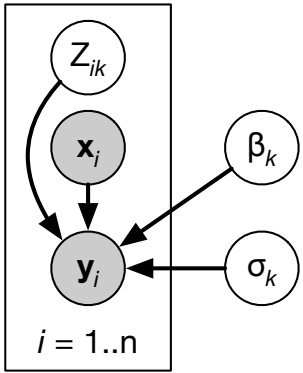
How many clusters? What is the distribution?



Modelling a mixture of regression models



What is the data generation process?



Mix. of Regression data generation

1. Initialize β_k (the slope of each model) and σ_k (the spread within each model)
2. Randomly assign group membership, $Z_{ik} = 1$ iff a point i belongs to a group k.
3. Generate: $Y_i | X_i, Z_{ik} = 1, \mu_k \sim \mathcal{N}(X_i \beta_k, \sigma_k^2)$

What is the expected log-likelihood?

Expected log-likelihood (to maximize):

$$\mathcal{L} = \sum_{i=1}^{n} \sum_{k=1}^{K} p(Z_{ik}|Y_i, X_i, \beta_k, \sigma_k) \log p(Y_i|X_i, \beta_k, \sigma_k)$$

where

$$\log p(Y_i|X_i, \beta_k, \sigma_k) = \log \mathcal{N}(Y_i|X_i\beta_k, \sigma_k^2) = -\frac{1}{2\sigma_k^2}(Y_i - X_i\beta_k)^2 - \frac{1}{2}\log \sigma_k^2$$

So, it's equivalent to finding weighted least square estimates (if $\sigma_k = 1$):

$$\min \sum_{k=1}^{K} \sum_{i=1}^{n} \underbrace{\mathbb{E}[Z_{ik}]}_{\text{E-step}} \underbrace{(Y_i - X_i\beta_k)^2}_{\text{M-step}}$$

E-step: What is the posterior probability of assigning each data point?

Given β_k and σ_k:

$$p(Z_{ik}|X_i, Y_i, \beta_k, \sigma_k) = \frac{\exp\left(-\frac{1}{2\sigma_k^2}\left(Y_i - \overbrace{X_i\beta_k}^{\text{prediction}}\right)^2 - \log\sigma_k\right)}{\sum_{k'}\exp\left(-\frac{1}{2\sigma_{k'}^2}\left(Y_i - \underbrace{X_i\beta_{k'}}_{\text{prediction}}\right)^2 - \log\sigma_{k'}\right)}$$

- Intuition: For the observed (X_i, Y_i), we ask, "how far is the predicted $X_i\beta_k$ from the observed Y_i?"
- Sample inversely proportional to the distance from different $X_i\beta_k$
- More rigorously: We need to introduce a Lagrangian multiplier to enforce the "sum to 1" constraint, $\sum_k Z_{ik} = 1$.

E-step: What is the posterior probability of assigning each data point?

Given β_k and σ_k:

$$p(Z_{ik}|X_i, Y_i, \beta_k, \sigma_k) = \frac{\exp\left(-\frac{1}{2\sigma_k^2}\left(Y_i - X_i\beta_k\right)^2 - \log\sigma_k\right)}{\sum_{k'}\exp\left(-\frac{1}{2\sigma_{k'}^2}\left(Y_i - X_i\beta_{k'}\right)^2 - \log\sigma_{k'}\right)}$$

- Intuition: For the observed (X_i, Y_i), we ask, "how far is the predicted $X_i\beta_k$ from the observed Y_i?"
- Sample inversely proportional to the distance from different $X_i\beta_k$
- More rigorously: We need to introduce a Lagrangian multiplier to enforce the "sum to 1" constraint, $\sum_k Z_{ik} = 1$.

M-step: Maximize regression model parameters

For each regression model k, we can optimize the parameters β_k and σ_k.

For example,

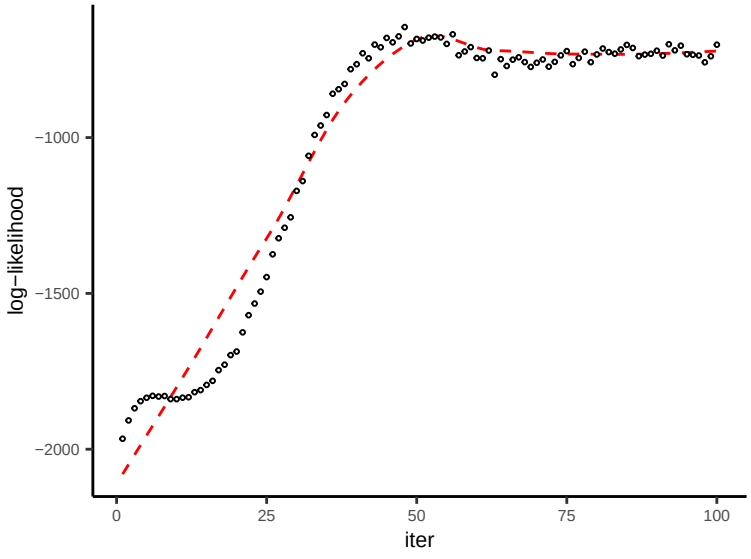
$$\nabla_{\beta_k,\sigma_k}\mathcal{L} = \overbrace{\sum_{i=1}^{n}}^{\text{could be a minibatch}} \underbrace{\mathbb{E}[Z_{ik}]}_{\text{from the E-step}} \underbrace{\nabla_{\beta_k,\sigma_k}\log p(Y_i|X_i,\beta_k,\sigma_k)}_{\text{local gradients}}$$

we can simply take stochastic gradient steps:

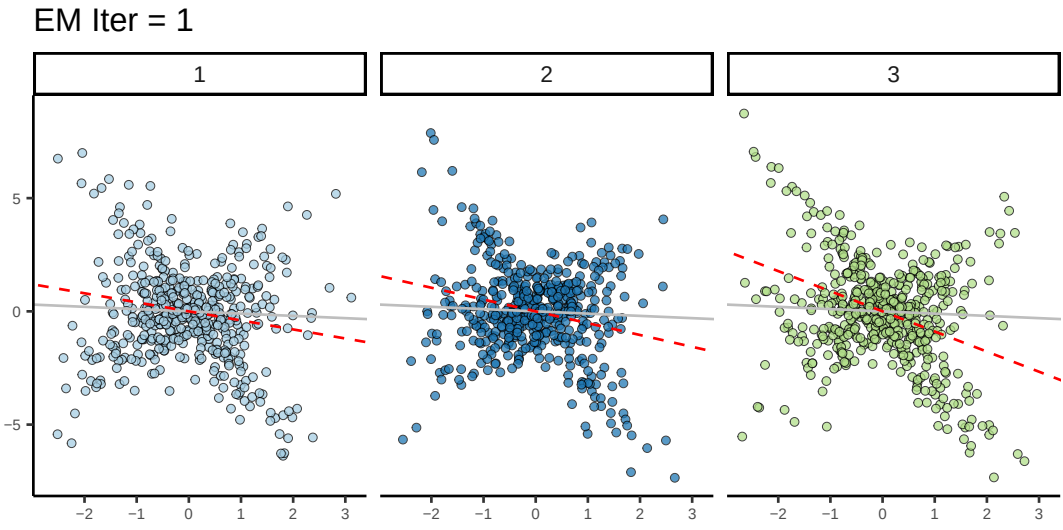
$$\beta_k \leftarrow \beta_k + \rho\nabla_{\beta_k}\mathcal{L}$$

$$\sigma_k \leftarrow \sigma_k + \rho\nabla_{\sigma_k}\mathcal{L}$$

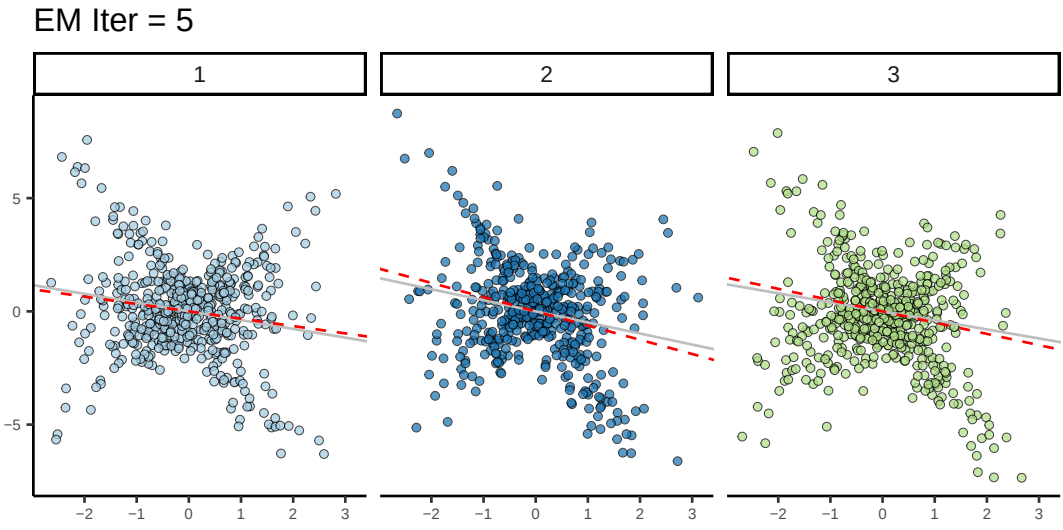
EM algorithm for a mixture of regression models



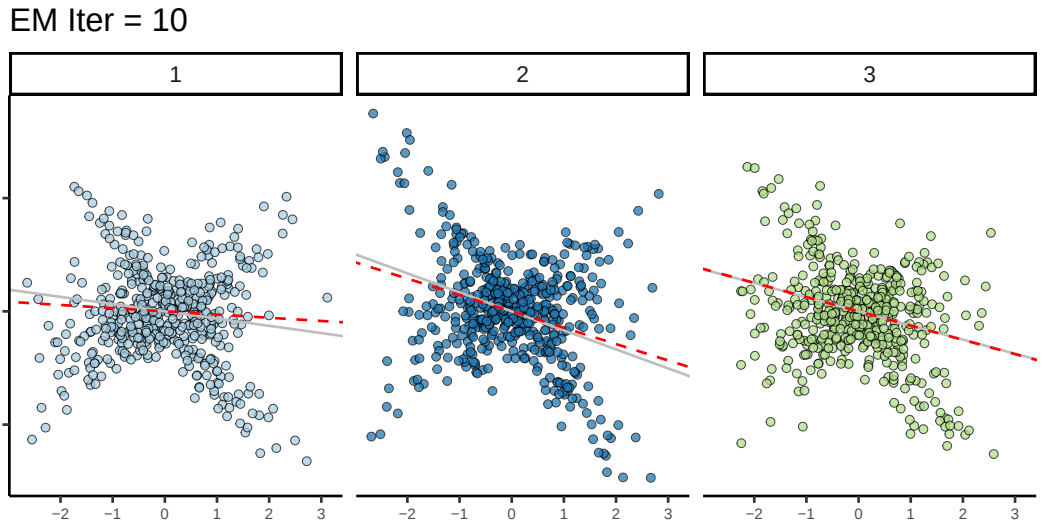
EM algorithm for a mixture of regression models



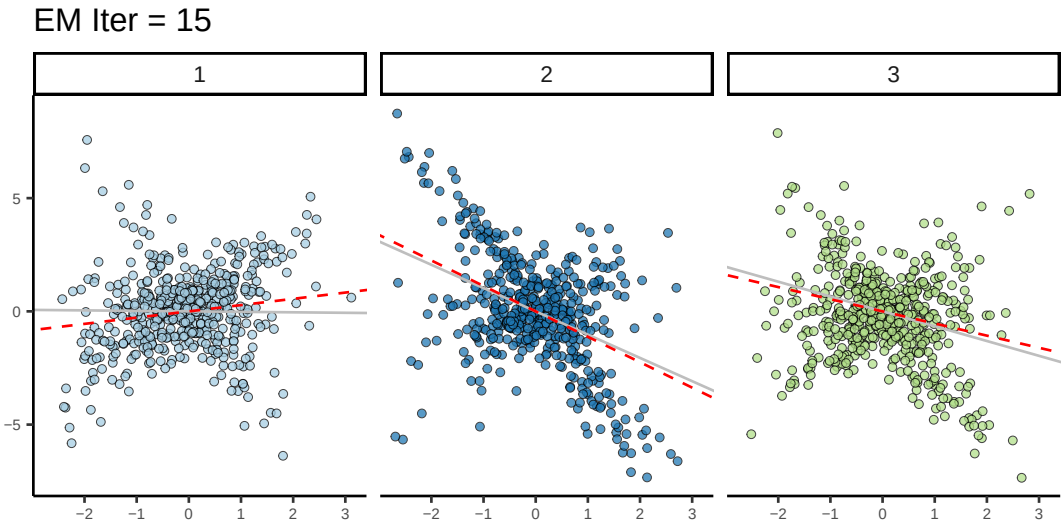
EM algorithm for a mixture of regression models



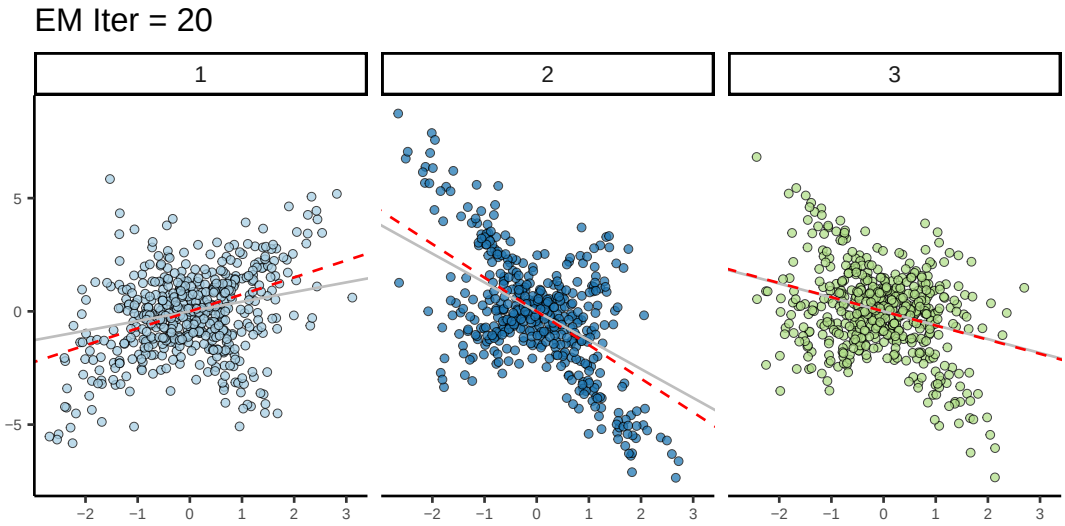
EM algorithm for a mixture of regression models



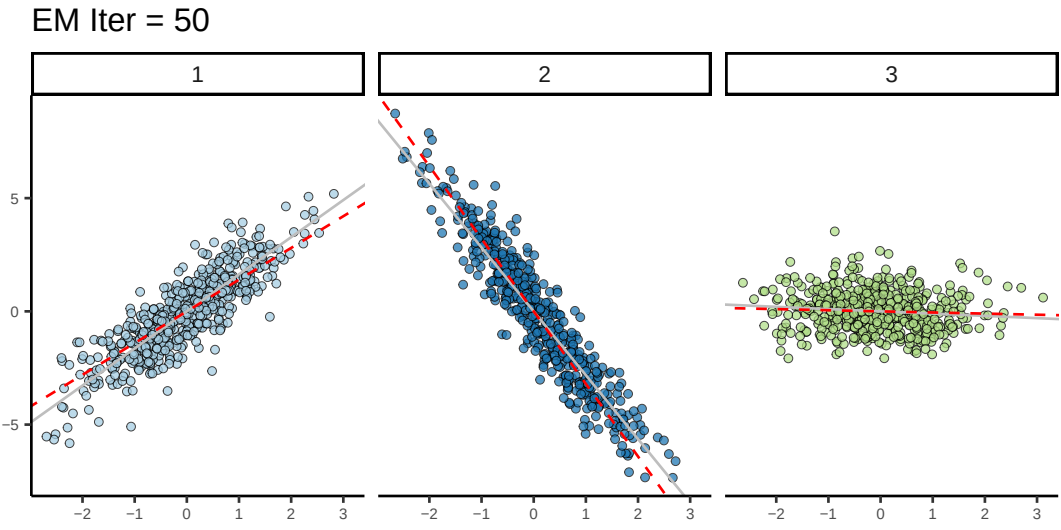
EM algorithm for a mixture of regression models



EM algorithm for a mixture of regression models



EM algorithm for a mixture of regression models



Discussions: a Mixture of Regression models

- What will be a potential application of this approach?
- Can we apply the same inference algorithm for non-linear regression models?
- What is the benefit of modelling a mixture of regressions models, as opposed to modelling a mixture of densities?

What you will learn

General discussions on unsupervised learning

Principal Component Analysis

Clustering to uncover common features and hidden groups

Advanced model-based clustering

Advanced latent variable modelling

Summary

Representation learning: How do we take advantage of both aspects?

Feature engineering

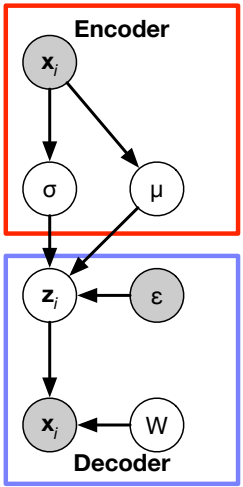
- Combine samples based on similarity to learn common features
- Combine features/genes/proteins to create a new factor
- Matrix factorization to capture factors that explain large variation

Manifold learning

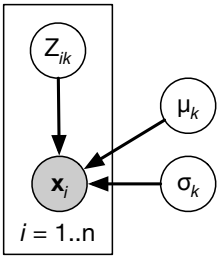
- Change the coordinate system
- What's not separable in one space can be spread out in another one
- Apply non-linear functions to a set of features

Variational Autoencoder

VAE



GMM



- Unlike traditional latent variable models, variational autoencoder models have the encoding layers to facilitate the latent variable inference.

Variational Autoencoder Model

Non-linear matrix factorization:

$$\mathbb{E}[\mathbf{x}_i|\mathbf{z}_i, \Theta] = \overset{\text{non-linear function}}{f}\left(\overset{\text{latent variable}}{\mathbf{z}_i} \quad \underset{\text{model parameter}}{\Theta} + \epsilon\right)$$

where $\mathbf{x}_i \in \mathbb{R}^V$ and $\mathbf{z}_i \in \mathbb{R}^K$

We can solve the maximum likelihood:

$$\log \sum_i \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} \geq \sum_i \mathbb{E}[\mathbf{z}_i|\cdot] \log p(\mathbf{x}_i|\Theta)$$

by Expectation Maximization updates

- E-step: sample or approximate $p(\mathbf{z}_i|\mathbf{x}_i, \Theta)$ **Usually, very hard**
- M-step: $\arg\max_{\Theta} \sum_i \mathbb{E}[\mathbf{z}_i]\, p(\mathbf{x}_i|\mathbf{z}_i, \Theta)$

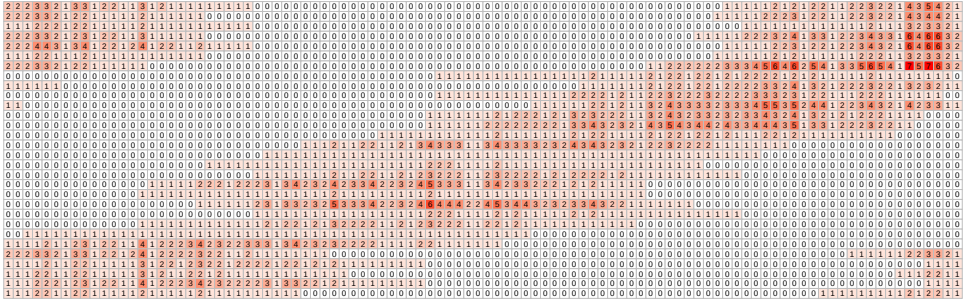
Learning latent structures from non-negative counting data

Non-linear counting data:

$$X_{ij} \leftarrow f\left(\sum_{k=1}^{3} Z_{ik} W_{kj}\right)$$

where $f(x) = \lceil \max\{x, 0\} \rceil$, and $Z_{ik}, W_{kj} \sim \mathcal{N}(0, 1)$.

sample x dimension



(a side note) How do we generate random numbers from a certain distribution?

Using CDF

- Cumulative Density Function: $F(z;\theta)$
- $\epsilon \sim U(0,1)$
- $F^{-1} : \epsilon \to z$

One-liner functions

- Exponential $z \sim \text{Exp}(z|\lambda)$:
  $\epsilon \sim U(0,1)$ $z = -\ln(\epsilon)/\lambda$
- Univariate Gaussian $\mathcal{N}(z|\mu,\sigma^2)$:
  $\epsilon \sim \mathcal{N}(0,1)$, $z = \mu + \sigma\epsilon$
- Multivariate Gaussian $\mathcal{N}(\mathbf{z}|\mu, RR^\top)$:
  $\epsilon \sim \mathcal{N}(0,I)$, $\mathbf{z} = \mu + R\epsilon$

Encoder (E-step): How do we approximate the posterior prob. of latent variables?

Let's directly model $p(\mathbf{z}_i|\mathbf{x}_i) \approx q(\mathbf{z}_i|\mathbf{x}_i)$ **a neural network** model!

For each latent variable k:

$$q(z_{ik}|\mathbf{x}) \sim \mathcal{N}(\mu_{ik}(\mathbf{x}_i), \exp(\ln \sigma^2_{ik}(\mathbf{x}_i)))$$

where

$$\mu_{ik}(\mathbf{x}_i) \leftarrow g_\mu(\mathbf{x}_i)$$
$$\ln \sigma^2_{ik}(\mathbf{x}_i) \leftarrow g_\sigma(\mathbf{x}_i)$$

Initially the encoder model will generate something random

$\mu(x)$

0.1	0.1	−0.2
−0.1	0.6	−0.2
0	0.1	0.2
−0.2	0.5	−0.2
−0.2	0.4	0.1
0	0.5	−0.2
−0.1	0.6	−0.6
−0.9	−0.1	0.2
−0.5	0.3	−0.3
−1.2	−0.1	−0.1
−0.5	0.1	−0.2
−1.3	0.1	0.1
−1.8	0	−0.1
−0.9	−0.2	−0.1
−1.6	−1.4	0.2
−0.6	−0.4	−0.1
−0.4	−0.7	0.3
−0.8	−0.9	0
0	−0.8	1
−0.1	−0.4	0.4
−0.5	−1.4	0.9
−0.7	−0.5	0
0	−0.3	0.6
0.1	0	0.2
0.8	0.2	0.9
0.1	0.4	0.1
0.7	0.3	0.5
0.4	0	0
0.6	0.1	0.5
0.2	0.2	0.2

$\log\sigma^2(x)$

1.5	−1	0.6
1.4	−0.9	0.3
1.2	−0.5	0.5
2.1	−0.9	0.3
2	−0.8	0.4
0.9	−0.7	0.3
2.9	−1.4	1
1.5	−0.3	0.5
2	−0.5	0.5
1.9	−0.5	0.7
2.8	−1	0.9
2.4	−0.7	0.8
3.1	−0.4	1.1
1.4	−0.3	0.7
1.7	0.3	0.9
0.5	−0.1	0.5
0.5	0.1	0.3
0.9	0	0.6
0.7	0.1	1
0.6	0.1	0.5
1.2	0.5	0.8
0.5	−0.1	0.3
0.5	0.2	0.8
0.1	−0.3	0.4
0.3	−0.5	0.4
0.7	−0.9	0.1
0.4	−0.5	0.4
0.5	−0.7	0.1
0.5	−0.8	0.4
0.6	−0.5	0.2

$z \sim N\left(\mu;\sigma^2\right)$

−2.6	0.3	0.3
1.3	2	−0.2
4	0.3	−0.7
−1.2	1.2	−2.3
4.7	−0.1	−2.9
0.8	−0.1	−0.9
4.3	−0.1	−1.8
2.2	−0.6	−2
2.6	1.6	−0.4
−4.2	−0.1	0.9
6.6	−1	0.1
2	0.5	0.2
−1.2	0.3	−2.1
−4.5	−0.2	2
−3.2	−1.9	3.4
−1.9	−2	0.3
−0.6	0.4	−0.9
0.3	−1.3	2.1
1.1	0.9	4.3
0.3	−1.5	−1.3
−1.1	−1.4	1.8
−0.2	−0.2	0.3
−0.6	−0.2	1.8
1.4	0.3	0.5
2	0.1	2.4
−1.2	−0.1	1.2
0	−0.2	1.3
1.4	−0.6	−0.2
1.6	0.1	0.7
−0.4	−0.2	−0.9

Decoder (M-step): Modeling data-generating process

Model X as Poisson with the mean parameter:

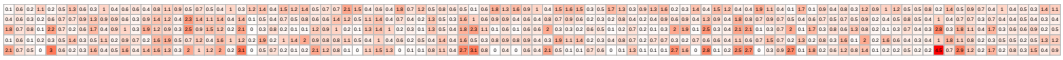
$$\mathbb{E}\left[X_{ij}|\mathbf{z}_i\right] = \ln\left(1+\exp\left(\sum_k Z_{ik}W_{kj}\right)\right)$$

Given latent state:

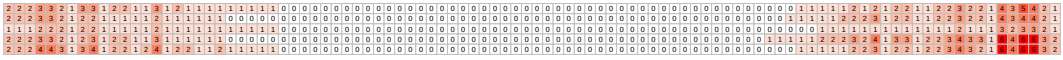
$Z=$

−2.6	0.3	0.3
1.3	2	−0.2
4	0.3	−0.7
−1.2	1.2	−2.3
4.7	−0.1	−2.9

Estimate mean $\lambda_{ij} \equiv \mathbb{E}\left[X_{ij}|Z_i\right]$

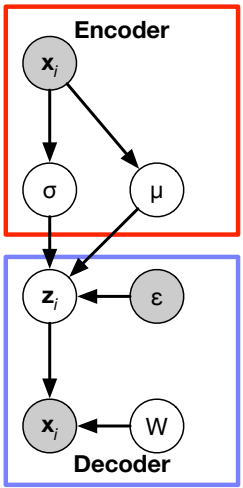


Observed data X



Variational Autoencoder

VAE

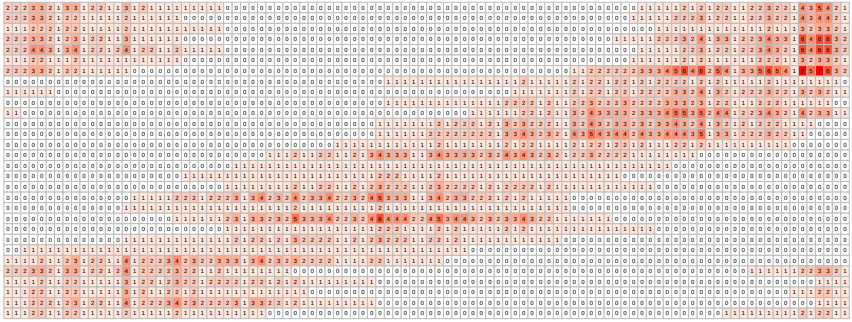


In practice:

- Define relationships between variables (auto generative process)
- Usually, the decoder side captures our scientific hypothesis
- We can use an "auto-diff" algorithm (e.g., Facebook `torch` or Google `tensorflow`) to calculate gradients for the model parameters to optimize.

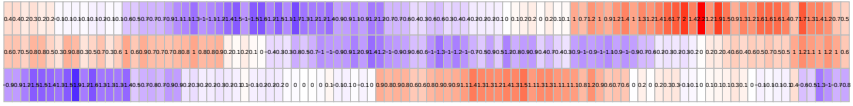
From the VAE parameters, we can extract a “dictionary” matrix and latent group structure

data

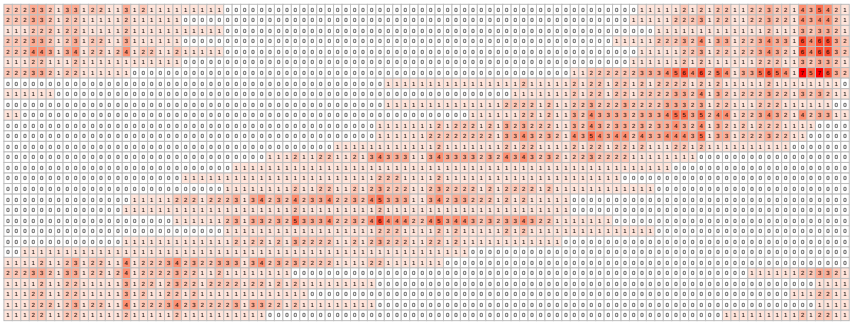


From the VAE parameters, we can extract a “dictionary” matrix and latent group structure

model parameters

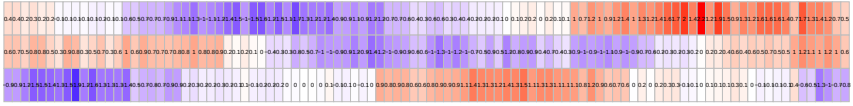


data

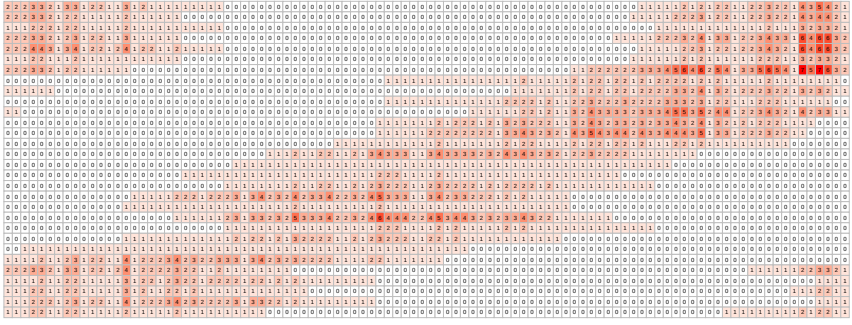


From the VAE parameters, we can extract a “dictionary” matrix and latent group structure

model parameters



data



Z

0.8	1.8	−1.1
1.2	1	−1.2
0.4	1.6	−0.7
1.7	1.2	−1.4
1	2.1	−1.4
0.9	0.8	−1
3.1	0.3	−1.2
1.2	−0.1	1.5
2	0.3	0.4
1.7	−0.6	1.4
2.7	0.1	1.1
1.8	−0.8	1.8
2.2	−1.2	2
0.9	−1.3	0.9
−0.2	−2.3	1.5
−0.4	−1.2	0.6
−1.1	−0.4	1.2
−0.8	−1.6	0.9
−2.9	−0.7	1.2
−1.7	0.2	1
−2.4	−1.6	1.8
−0.7	−1.3	0.8
−2.1	−0.4	0.8
−1.5	−0.6	−0.9
−2.7	0.1	−1.4
−0.7	2	−1.3
−1.7	0.5	−1.5
−1	1.4	−1
−2	0.6	−1.7
−0.4	1.9	−0.5

Other types of latent variable methods

(We will cover them in the genetics and single-cell lectures)

Specialized latent variable models

- Hidden Markov Model
- Spatially-constrained linear models
- Admixture models (genetics)
- Embedded Topic Models

Distance-based learning/embedding

- t-Stochastic Neighbourhood Embedding (tSNE)
- Uniform Manifold Approximation & Projection (UMAP)
- Hierarchical agglomerative clustering

What you will learn

General discussions on unsupervised learning

Principal Component Analysis

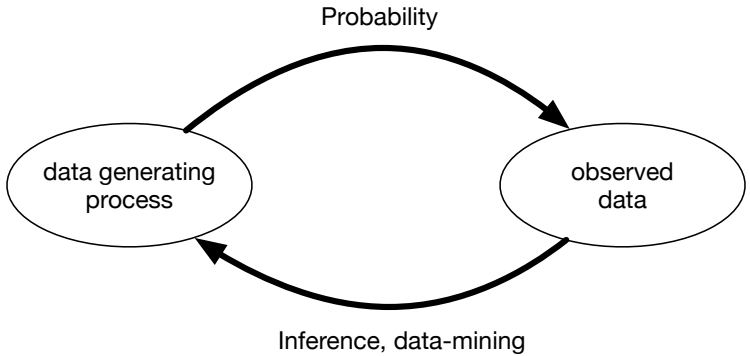
Clustering to uncover common features and hidden groups

Advanced model-based clustering

Advanced latent variable modelling

Summary

Discussions



Wassermann, *All of Statistics* (2004)