Machine Learning 1.13: Latent Variables

Tom S. F. Haines
T.S.F.Haines@bath.ac.uk

Knowing Everything

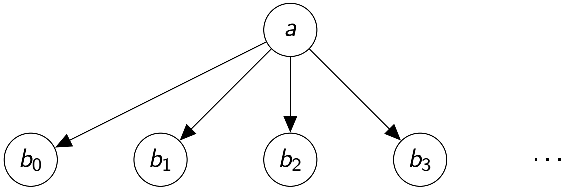
- Previous two lectures: We know everything! (during training)
- This lecture: We don't.

Knowing Everything

- Previous two lectures: We know everything! (during training)
- This lecture: We don't.

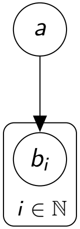
- Structure often unknown, incomplete, or a reasonable guess.
- Don't have all latent RVs when training (Often hypothetical RVs).

Aside: Plate Notation



- Often need to represent repetition.
- Tree becomes confusing quickly.

Aside: Plate Notation

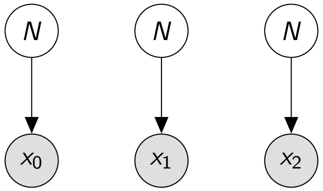


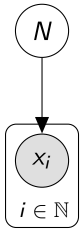
- Often need to represent repetition.
- Tree becomes confusing quickly.

- **Plate notation** solves this.
- Note indices.
- Can cover multiple RVs.

One Model

- Shared parameters can be made explicit using plates:





(e.g. Gaussian distribution N over x)

- Much easier for complex models.
- Important for fully Bayesian.

Examples

Two examples:

- Latent Semantic Analysis (recommender system).
- Gaussian Mixture Model (density estimation).

Example I: Recommender System

- A type of search engine.
- Estimates what you will like/dislike.

- Amazon, Netflix etc. all use this to recommend products.

Example I: Recommender System

- A type of search engine.
- Estimates what you will like/dislike.

- Amazon, Netflix etc. all use this to recommend products.

- Two approaches:
  - Content-based filtering:
    Each product has features. Match user to products with features similar to previously liked (e.g. per user logistic regression model).
  - Collaborative filtering:
    Identify users with similar likes. Assume they have similar opinions for products where one has given an opinion and the other has not (e.g. clustering users).

  Hybrids often used in practise.

Latent Semantic Analysis

- Also called *Latent Semantic Indexing*.
- Mathematically similar to *Principal Component Analysis*.

Latent Semantic Analysis

- Also called *Latent Semantic Indexing*.
- Mathematically similar to *Principal Component Analysis*.

- Usually described as though it's for text analysis only.
- Recommender systems people often refer to this as *SVD*.
  (as in the matrix factorisation – seriously stupid name)

Latent Semantic Analysis

- Also called *Latent Semantic Indexing*.
- Mathematically similar to *Principal Component Analysis*.

- Usually described as though it's for text analysis only.
- Recommender systems people often refer to this as *SVD*.
  (as in the matrix factorisation – seriously stupid name)

- Has multiple equivalent graphical models!
- **Not probabilistic** (though easy to fix).

Unknown Attributes

- Have a set of unknown attributes.
- Each user likes or dislikes each attribute with a strength.
- Each film has the attributes, with various strengths.

Unknown Attributes

- Have a set of unknown attributes.
- Each user likes or dislikes each attribute with a strength.
- Each film has the attributes, with various strengths.

- Seems reasonable:
  e.g. Robert likes action films. Star Wars is an action film.
  ∴ Robert likes Star Wars.
- With multiple attributes:
  e.g. Susan likes action films but really hates fantasy. Star Wars is a fantasy action film.
  ∴ Susan dislikes Star Wars a little bit.

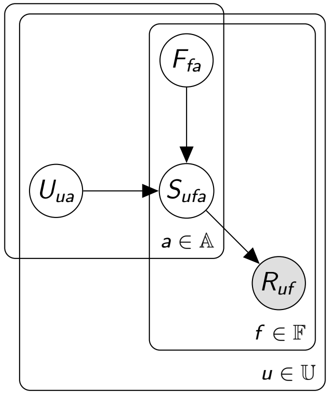
Unknown Attributes

- Have a set of unknown attributes.
- Each user likes or dislikes each attribute with a strength.
- Each film has the attributes, with various strengths.

- Seems reasonable:
  e.g. Robert likes action films. Star Wars is an action film.
  $\therefore$ Robert likes Star Wars.
- With multiple attributes:
  e.g. Susan likes action films but really hates fantasy. Star Wars is a fantasy action film.
  $\therefore$ Susan dislikes Star Wars a little bit.

- **Unknown** $\therefore$ Inferred from data.
- Unlikely to have a human interpretation. Does this matter?

Graphical Model



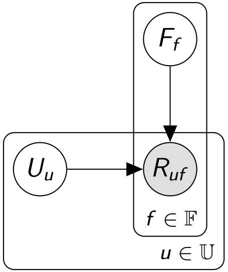
- a – Index of an attribute, $\in \mathbb{A}$.
- u – Index of a user, $\in \mathbb{U}$.
- f – Index of a film, $\in \mathbb{F}$.

- F_{fa} – Strength of attribute a in film f.
- U_{ua} – How much user u likes attribute a.

- S_{ufa} – Score: Effect of attribute a for user u on film f (multiplication).
- R_{uf} – Rating given by user u of film f (addition).

- Training data: **Only** R_{uf}, extremely sparse.

Simplified



- You usually simplify by not showing the attributes. (the arrows are now vectors of attribute weights)
- This is also the standard PCA graphical model.

Fitting to Data

Analytic:

1. Calculate A, where $A_{uf} = R_{uf}$ when known, zero otherwise.

Fitting to Data

Analytic:

1. Calculate A, where $A_{uf} = R_{uf}$ when known, zero otherwise.
2. Perform SVD, obtain $A = U\Sigma V^T$. Assume Σ (singular values) sorted high to low.

Fitting to Data

Analytic:

1. Calculate A, where $A_{uf} = R_{uf}$ when known, zero otherwise.
2. Perform SVD, obtain $A = U\Sigma V^T$. Assume Σ (singular values) sorted high to low.
3. Keep the top N columns of U and V, where N is the number of attributes (hyper-parameter).

Fitting to Data

Analytic:

1. Calculate A, where $A_{uf} = R_{uf}$ when known, zero otherwise.
2. Perform SVD, obtain $A = U\Sigma V^T$. Assume Σ (singular values) sorted high to low.
3. Keep the top N columns of U and V, where N is the number of attributes (hyper-parameter).
4. Scale column n by $\sqrt{\Sigma_n}$ (U and V).

U contains the attribute weights for users; V the attribute weights for films.

Fitting to Data

Analytic:

1. Calculate A, where $A_{uf} = R_{uf}$ when known, zero otherwise.
2. Perform SVD, obtain $A = U\Sigma V^T$. Assume Σ (singular values) sorted high to low.
3. Keep the top N columns of U and V, where N is the number of attributes (hyper-parameter).
4. Scale column n by $\sqrt{\Sigma_n}$ (U and V).

U contains the attribute weights for users; V the attribute weights for films.

In practise:
Too much data, so use gradient descent to minimise error instead.

Results

From the Netflix challenge:

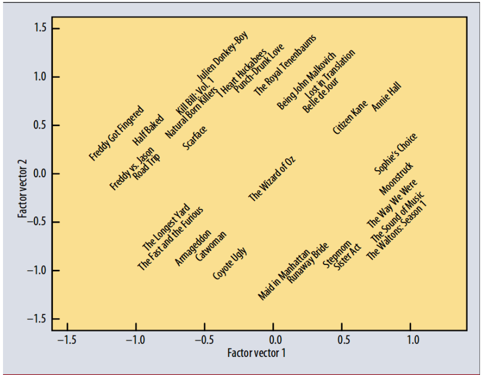


Figure 3. The first two vectors from a matrix decomposition of the Netflix Prize data. Selected movies are placed at the appropriate spot based on their factor vectors in two dimensions. The plot reveals distinct genres, including clusters of movies with strong female leads, fraternity humor, and quirky independent films.

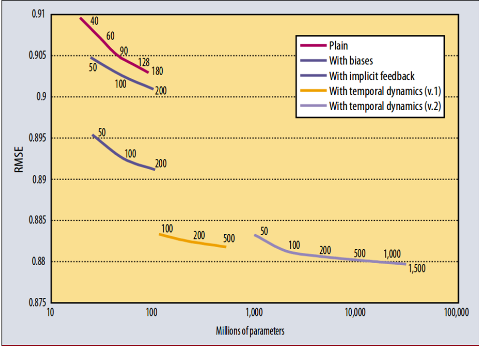


Figure 4. Matrix factorization models' accuracy. The plots show the root-mean-square error of each of four individual factor models (lower is better). Accuracy improves when the factor model's dimensionality (denoted by numbers on the charts) increases. In addition, the more refined factor models, whose descriptions involve more distinct sets of parameters, are more accurate. For comparison, the Netflix system achieves RMSE = 0.9514 on the same dataset, while the grand prize's required accuracy is RMSE = 0.8563.

Latent Semantic Analysis Notes

- PCA without subtracting the means, trained with sparse data.
- Subtracting means recommended however!

Latent Semantic Analysis Notes

- PCA without subtracting the means, trained with sparse data.
- Subtracting means recommended however!

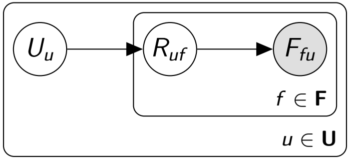
- Needs lots of data – does badly for users/films with few ratings (cold start problem).
- Standard probabilistic variant not suitable for recommender systems.

Latent Semantic Analysis Notes

- PCA without subtracting the means, trained with sparse data.
- Subtracting means recommended however!

- Needs lots of data – does badly for users/films with few ratings (cold start problem).
- Standard probabilistic variant not suitable for recommender systems.

Normal graphical model – weird conditional structure for this problem:

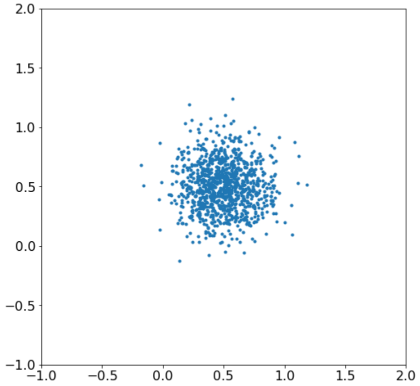


(user $\rightarrow$ document, rating $\rightarrow$ topic, film $\rightarrow$ word)

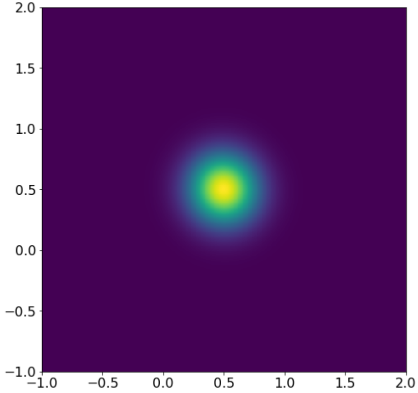
Example II: Density Estimation

- Density estimation:
  - Input: Samples, x, drawn from an unknown probability distribution D.
  - Output: An estimate of D.

Input:



Output: (Gaussian)



Example II: Density Estimation

- Density estimation:
  - Input: Samples, x, drawn from an unknown probability distribution D.
  - Output: An estimate of D.

- Examples:
  - Histogram (normalised)
  - Fitting a Gaussian
  - Gaussian Mixture Model (GMM)

Example II: Density Estimation

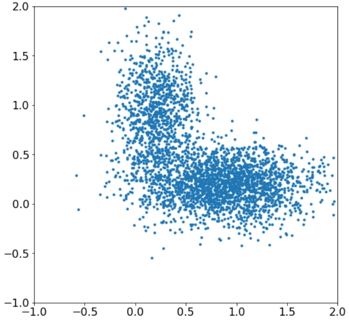
- Density estimation:
    - Input: Samples, x, drawn from an unknown probability distribution D.
    - Output: An estimate of D.
- Examples:
    - Histogram (normalised)
    - Fitting a Gaussian
    - Gaussian Mixture Model (GMM)
- Uses:
    - Visualisation.
    - Clustering.
    - Regression/classification via Bayes rule.
    - Abnormality detection.
    - Part of larger models, e.g. voice recognition.

Gaussian Mixture Model

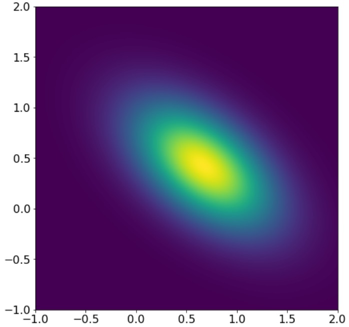
- Instead of one Gaussian distribution describe data with several.

$$P(x|\alpha, \mu, \Sigma) = \sum_{z=1}^{K} \alpha_z \mathcal{N}(x|\mu_z, \Sigma_z)$$

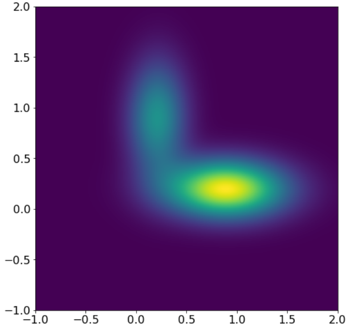
Input:



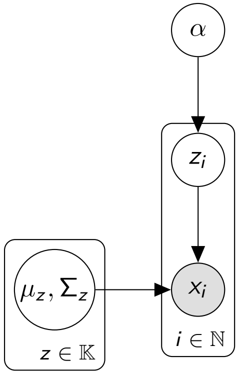
Gaussian:



GMM with 2 components:



Graphical Model



- i – Index of exemplar, $\in \mathbb{N}$.
- z – Index of component, $\in \mathbb{K}$.
- μ_z, Σ_z – Mean and covariance for component z.
- α – Categorical distribution over component membership.
- z_i – Which component feature vector i belongs to.
- x_i – Feature vector for exemplar i.

Fitting to Data

- No analytical solution.
- Belief propagation problematic.

- New algorithm: **Expectation Maximisation**

Expectation Maximisation

- Finds maximum likelihood and maximum a posteriori solutions.
- Similar to gradient descent:
  - Start from random parameters.
  - Take a series of steps that improve model fit to data.
  - Get stuck in local minima.

Expectation Maximisation

- Finds maximum likelihood and maximum a posteriori solutions.
- Similar to gradient descent:
  - Start from random parameters.
  - Take a series of steps that improve model fit to data.
  - Get stuck in local minima.

- Divide graphical model into three parts:
  - **Model parameters**: Latent variables shared between all data.
  - **Missing values**: Latent variables associated with data. (Often called latent variables!)
  - **Observed variables**: The data.

Expectation Maximisation

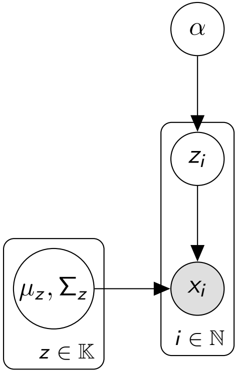
- Finds maximum likelihood and maximum a posteriori solutions.
- Similar to gradient descent:
    - Start from random parameters.
    - Take a series of steps that improve model fit to data.
    - Get stuck in local minima.

- Divide graphical model into three parts:
    - **Model parameters**: Latent variables shared between all data.
    - **Missing values**: Latent variables associated with data. (Often called latent variables!)
    - **Observed variables**: The data.

- Two steps:
    - **E-step**: Calculate **likelihood function** of missing values with model parameters fixed. (Often referred to as calculating **expectation**. It isn't.)
    - **M-Step**: **Maximise** model parameters given expectation of missing values.

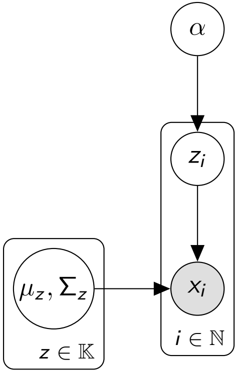
  Alternate between until convergence.

EM for GMM



- Model parameters: α, μ, Σ
- Missing values: z
- Observed variables: x

EM for GMM



- Model parameters: α, μ, Σ
- Missing values: z
- Observed variables: x

- Likelihood of z_i:
  Categorical distribution, $z_i \sim \texttt{Cat}(w_i)$
  $\sum_{z=1}^{K} w_{iz} = 1$

Initialisation

- Any random parameters will do, but. . .

Initialisation

- Any random parameters will do, but. . .
- Much better:
  1. Run k-means with K centres.
  2. Set $\alpha \propto$ number assigned to each centre.
  3. Fit μ and Σ to points assigned to each centre.
- This is close to the goal $\therefore$ less iterations required to converge.
- There are clever k-means initialisations too.

E-Step

Set (for all i, z):

$$w_{iz} \propto \alpha_z \mathcal{N}(x_i | \mu_z, \Sigma_z)$$

(remember to normalise, so $\sum_{z=1}^{K} w_{iz} = 1$)

M-Step

Set (for all z):

$$\alpha_z \propto T_z,\, T_z = \sum_{i=1}^{N} w_{iz}$$

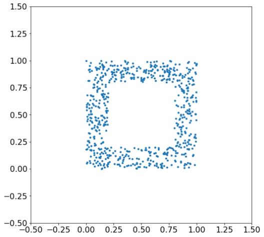
(remember to normalise, so $\sum_{z=1}^{K} \alpha_z = 1$)

$$\mu_z = \frac{1}{T_z} \sum_{i=1}^{N} w_{iz} x_i$$

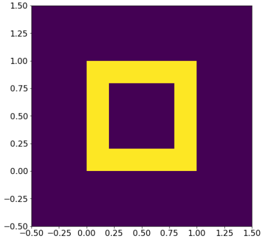
$$\Sigma_z = \frac{1}{T_z} \sum_{i=1}^{N} w_{iz} (x_i - \mu_z)(x_i - \mu_z)^T$$

Demonstration

Input:

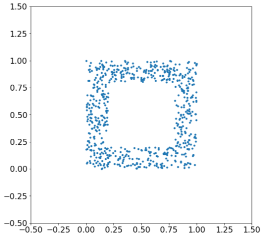


Ground Truth:

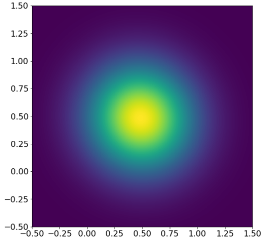


Demonstration

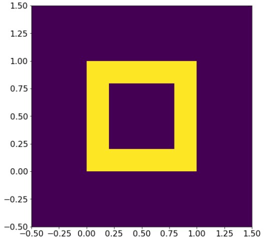
Input:



Gaussian:

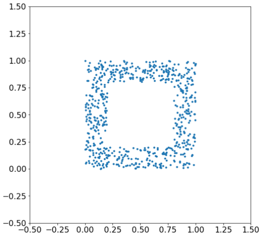


Ground Truth:

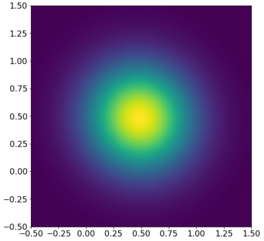


Demonstration

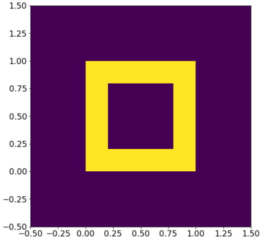
Input:



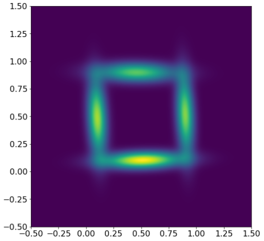
Gaussian:



Ground Truth:

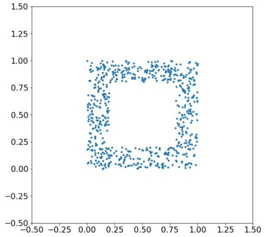


GMM with 4 components:

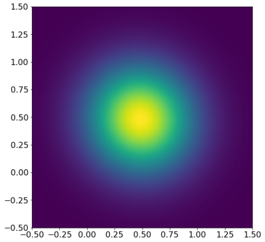


Demonstration

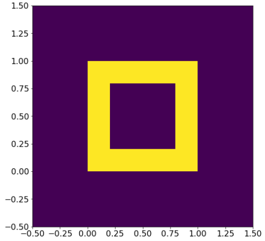
Input:



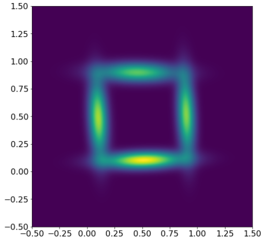
Gaussian:



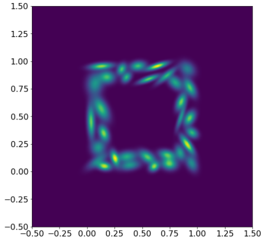
Ground Truth:



GMM with 4 components:



GMM with 32 components:



How Many Components?

- Choosing K can be hard.

- Alternatively:
  1. Fit many models, with different component counts.
  2. Calculate *Bayesian information criterion* for each.
  3. Choose best.

How Many Components?

- Choosing K can be hard.

- Alternatively:
  1. Fit many models, with different component counts.
  2. Calculate *Bayesian information criterion* for each.
  3. Choose best.

- BIC is crude – poor assumptions. Many other options.
- Non-parametric techniques, e.g. Dirichlet process prior ideal.

Summary

- Real problems are not neat.
- Plate notation.

- Latent Semantic Analysis.
- Gaussian Mixture Model.

Further Reading

- Paper describing the use of LSA on the Netflix challenge:
  "*Matrix Factorization Techniques for Recommender Systems*"
  by Koren, Bell & Volinsky (2009) (source of results images).

- Original EM paper, if you're so inclined:
  "*Maximum Likelihood from Incomplete Data via the EM Algorithm*",
  by Dempster, Laird, & Rubin (1977).
  (Special cases were discovered before, has mistake in convergence proof)

- If you sensibly decide to ignore the above:
  Chapter 9 of "*Pattern Recognition and Machine Learning*" by Bishop.