Lecture 1

Introduction - Part 2

Luigi Freda

ALCOR Lab
DIAG
University of Rome "La Sapienza"

October 10, 2016

Outline

Outline

Unsupervised Learning

Unsupervised learning

- in this case, the training set is $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N$ and the goal is to find "interesting patterns" in it
- this is sometimes called **knowledge discovery**
- unlike supervised learning, we are not told what the desired output is for each input

Unsupervised Learning

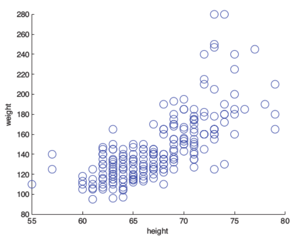
Unsupervised learning

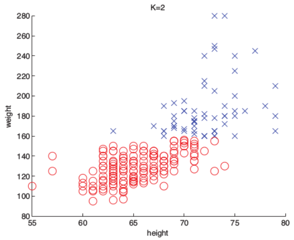
- can be formalized as a **density estimation**, that is, we want to build models of the form $p(\mathbf{x}_i|\theta)$
- θ is the parameter vector of the model
- there are two differences from the supervised case
  1. we have written $p(\mathbf{x}_i|\theta)$ instead of $p(y_i|\mathbf{x}_i, \theta)$; that is, supervised learning is conditional density estimation, whereas unsupervised learning is **unconditional density estimation**
  2. $\mathbf{x}_i$ is a vector of features, so we need to create **multivariate probability models**. By contrast, in supervised learning, y_i is usually just a single variable that we are trying to predict (this means that for most supervised learning problems, we can use univariate probability models)
- unsupervised learning more widely applicable than supervised learning, since it does not require a human expert to manually label the data

Outline

Discovering clusters

Discovering clusters - an example

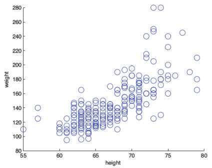


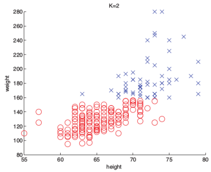


- consider the problem of clustering data into groups
- the above plot represents the height and weight of a group of 210 people
- let K denote the number of clusters
- man goal: estimate the distribution $p(K|D)$
- problem: we don't actually have an idea of the number of clusters

Discovering clusters

Discovering clusters





- first goal: estimate the distribution $p(K|D)$
- second goal: estimate which cluster each point belongs to

Some details

- **model selection**: we have to pick a model of the "right" complexity
- Let $z_i \in \{1, ..., K\}$ represents the cluster to which data point i is assigned (z_i is an example of a hidden or **latent variable**, never observed in the training set)
- we can compute $z_i = \underset{k}{\operatorname{argmax}}\; p(z_i = k|\mathbf{x}_i, \mathcal{D})$

Discovering clusters

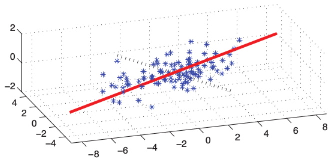
Discovering clusters - real world applications

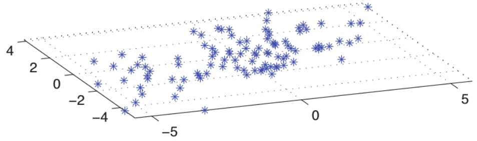
- **astronomy**, the autoclass system (Cheeseman et al. 1988) discovered a new type of star, based on clustering astrophysical measurements
- **e-commerce**, cluster users into groups, based on their purchasing or web-surfing behavior, and then send customized targeted advertising to each group (see e.g., (Berkhin 2006))
- **biology**, to cluster flow-cytometry data into groups, to discover different sub-populations of cells (see e.g., (Lo et al. 2009))

Outline

Discovering Latent Factors

Discovering latent factors - an example

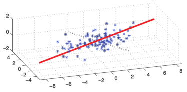


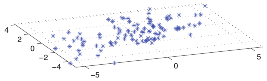


- **dimensionality reduction**: it is often useful to reduce the dimensionality by projecting the data to a lower dimensional subspace which captures the "essence" of the data
- for example in the above plot:
    - the 2d approximation is quite good, most points lie close to this subspace
    - projecting points onto the red line 1d approx is a rather poor approximation

Discovering Latent Factors

Discovering latent factors - motivations





- although data may appear high dimensional, there may only be a small number of degrees of variability, corresponding to **latent factors**
- low dimensional representations, when used as input to statistical models, often result in **better predictive accuracy** (focusing on the "essence")
- low dimensional representations are useful for **enabling fast** nearest neighbor **searches**
- two dimensional projections are very useful for **visualizing high dimensional data**

Discovering Latent Factors

Discovering latent factors - PCA

- the most common approach to dimensionality reduction is called **Principal Components Analysis** or **PCA**
- given $\mathbf{x}_i \in \mathbb{R}^D$, we compute an approximation $\hat{\mathbf{x}}_i = \mathbf{W}\mathbf{z}_i$ with
  - $\mathbf{z}_i \in \mathbb{R}^L$ where $L < D$ (dimensionality reduction)
  - $\mathbf{W} \in \mathbb{R}^{D \times L}$ and $\mathbf{W}^T\mathbf{W} = \mathbf{I}$ (orthogonal $\mathbf{W}$)

  so as to minimize the **reconstruction error**

$$J(\mathbf{W}, \mathbf{Z}) = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2 = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{W}\mathbf{z}_i\|^2$$

- this can be thought of as an unsupervised version of (multi-output) linear regression, where we observe the high-dimensional response $\mathbf{y} = \mathbf{x}$, but not the low-dimensional “cause” $\mathbf{z}$

Discovering Latent Factors

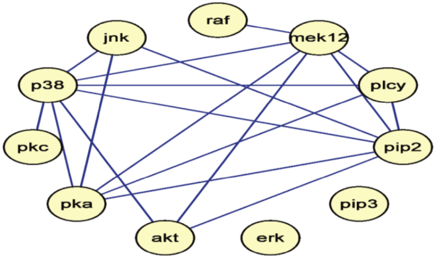
Discovering latent factors - PCA - real world applications

- **biology**, use PCA to interpret gene microarray data, to account for the fact that each measurement is usually the result of many genes which are correlated in their behavior by the fact that they belong to different biological pathways
- **signal processing** (e.g., of acoustic or neural signals), use **ICA** (which is a variant of PCA) to separate signals into their different sources
- **computer graphics**, project motion capture data to a low dimensional space, and use it to create animations

Outline

Discovering Graph Structure

Discovering graph structure



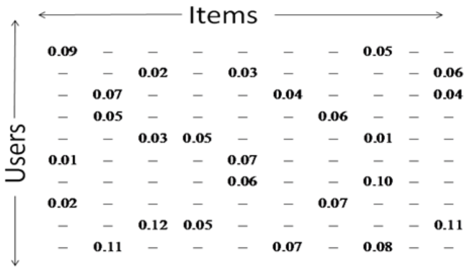
- sometimes we measure a set of **correlated variables**, and we would like to discover **which ones are most correlated** with which others
- this can be represented by a graph G: **nodes** represent variables, **edges** represent direct dependence between variables
- we can then learn this graph structure from data, i.e., we compute $\hat{G} = \mathrm{argmax} p(G|D)$

Outline

Matrix Completion

Matrix completion

A common situation: the design matrix has “holes” in it; the **goal** is to infer plausible values for the missing entries (”fill in” the holes)

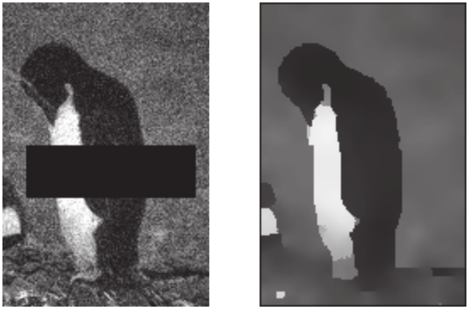

Items
Users

0.09	–	–	–	–	–	–	0.05	–	–
–	–	0.02	–	0.03	–	–	–	–	0.06
–	0.07	–	–	–	0.04	–	–	–	0.04
–	0.05	–	–	–	–	0.06	–	–	–
–	–	0.03	0.05	–	–	–	0.01	–	–
0.01	–	–	–	0.07	–	–	–	–	–
–	–	–	–	0.06	–	–	0.10	–	–
0.02	–	–	–	–	–	0.07	–	–	–
–	–	0.12	0.05	–	–	–	–	–	0.11
–	0.11	–	–	–	0.07	–	0.08	–	–

Matrix Completion

Examples

Image Inpainting



recall that computers internally store an image as a matrix data structure

Matrix Completion

Examples

Collaborative filtering

Alice	4			4
Bob		5	4	
Joe		5		
Sam	5			

- problem: predicting which movies people will want to watch based on how they, and other people, have rated movies which they have already seen
- key idea: prediction is not based on features of the movie or user (although it could be), but on a ratings matrix $\mathbf{X}$ where $\mathbf{X}(m, u)$ is the rating by user u of movie m
- most of the entries in $\mathbf{X}$ will be missing or unknown

Credits

- Kevin Murphy's book