Spatial Statistics with Image Analysis
Lecture 9

Johan Lindström

November 28, 2016

Column stacking

We use column-stacking of the image data, transforming back and forth between \( m \times n \times d \)-matrices and \( mn \times d \)-matrices.

\[
Y_{\text{col-stacked}} = \begin{bmatrix}
y_{1,1,1} & \cdots & y_{1,1,d} \\
y_{2,1,1} & \cdots & y_{2,1,d} \\
\vdots & \ddots & \vdots \\
y_{m,1,1} & \cdots & y_{m,1,d} \\
y_{1,2,1} & \cdots & y_{1,2,d} \\
\vdots & \ddots & \vdots \\
y_{m,n,1} & \cdots & y_{m,n,d}
\end{bmatrix}
\]
We have previously seen the entire image as a stochastic field, for classification we instead treat the pixels as independent vector valued random variables.

- Each pixel $y_i = y(s_i)$ is regarded as an observation of the same random variable.
- Each row in a column stacked image is an (independent) observation of $y$.
- The covariance of $y$ is then $\Sigma = \mathbb{V}(y)$.
- The covariance can be estimated as $\hat{\Sigma} = \frac{1}{N} (Y - \bar{Y})^T (Y - \bar{Y})$.
- Note that $\Sigma$ is now a $d \times d$ covariance matrix for the dependence between the colors (layers in the image).

Data reduction — Principal component analysis

- Do we really need all the dimensions (colour-layers) of the data?
- How can we extract the essential information?

Principal Component Analysis (PCA)

Diagonalise the covariance matrix, $\Sigma$:

$$\Sigma = P \Lambda P^T, \quad PP^T = P^T P = I,$$

$$\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_d), \quad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n > 0.$$ 

Then $yP_1$ is the linear combination of the $y$-components that has the largest variance.

This is also known as empirical orthogonal function (EOF).

The effect of the PCA transformation

$$C(y) = \Sigma = P \Lambda P^T$$

$$C(yP) = P^T C(y) P = P^T P \Lambda P^T P = \Lambda$$

- This also means that the transformed components are uncorrelated.
- It does not mean that the components are independent.
- Variance is not equivalent to information; The first principal components are not necessarily the most informative.
Let $y$ be a matrix with one $d$-dimensional observation per row, with the mean subtracted.

### Eigen Decomposition

Given an estimate of $\Sigma$, e.g. $\text{Sigma}=(y\,\times\,y)/\text{size}(y,1)$, compute the eigenvectors and eigenvalues of $\Sigma$ as $[P,\text{Lambda}]=\text{eig}($Sigma$)$, and sort the eigenvectors in decreasing eigenvalue order.

\[ \Sigma = P \Lambda P^\top, \quad PP^\top = P^\top P = I, \]
\[ \Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_d), \quad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \geq 0. \]

### Singular Value Decomposition

Compute the singular value decomposition of $y$ as $[U,S,V]=\text{svd}(y/\sqrt{\text{size}(y,1)}),0)$, where

\[ \frac{y}{\sqrt{N}} = USV^\top, \quad U^\top U = I, \]
\[ S = \text{diag}(s_1, s_2, \ldots, s_d), \quad V^\top V = I. \]

This option usually is numerical more stable.
Singular Value Decomposition

For the singular value decomposition of \( y \) we have that

\[
P \Lambda P^T = \Sigma = \frac{y^T y}{N} = VS^2 V^T.
\]

Identifying parts gives:

\[
\Lambda = S^2, \quad P = V, \quad \lambda_i = s^2_i.
\]

Object classification

We present object classification as an image segmentation problem, i.e. we want to classify all pixels in an image.

- We assume that there is a set of \( K \) object categories.
- We take the latent variables \( z_i = z(s_i) \) to be the (random) category for pixel \( y_i \) at location \( s_i \).
- The prior probabilities \( \pi_k \) describe the relative abundance of each object class.
- For each object class, there is a data model/distribution, \( p(y_i|z_i = k) \).

The same methodology can be applied to other kinds of data.

Object classification — Example

Given an image with pixels of 2 colours (green and blue) we want to classify each of the pixels into green and blue pixels respectively.
Object classification (cont.)

For now we assume that the prior categories of the pixels are independent in space

\[ p(z_1, z_2) = p(z_1)p(z_2). \]

with the same prior probability of belonging to a specific class everywhere

\[ p(z_i = k) = \pi_k, \quad \forall i. \]

Further we assume that the data distribution for the different classes follow different Normal distributions

\[ [y_i | z_i = k] \in N(\mu_k, \Sigma_k). \]

Bayesian classification

Since all the pixels are independent we now consider a single pixel \( y_i \) and its class belonging \( z_i \).

- Assume that \( \pi_k \) and the parameters of \( p(y_i|z_i = k), \mu_k \) and \( \Sigma_k \), for \( k = 1, \ldots, K \), are known.
- Given known parameters the posterior probabilities that a pixel belongs to class \( k \) is

\[ p(z_i = k | y_i) = \frac{p(y_i|z_i = k)p(z_i = k)}{p(y_i)} = \frac{p(y_i|z_i = k)\pi_k}{\sum_j p(y_i|z_i = j)\pi_j} \]

- Maximum A Posteriori classification assigns the object class with the highest posterior probability to each pixel.
- If all \( \Sigma_k \) are equal, we obtain the method Linear discriminant analysis. Otherwise, we obtain Quadratic discriminant analysis.

Bayesian classification

![Linear Bayesian discrimination](image1)

![Quadratic discrimination](image2)
Supervised classification — Training data

For supervised classification we have a (small) training set consisting of pixels with known class belonging, $z_i$.

Since we know the class belonging of each pixel in the training set estimates of the parameters are the standard mean and variance estimates based on independent samples.

\[
\hat{\pi}_k = \frac{\sum I(z_i = k)}{\sum_k \pi_k n_k}  \\
\hat{\mu}_k = \frac{1}{n_k} \sum_{i:z_i=k} y_i  \\
\hat{\Sigma}_k = \frac{1}{n_k - d} \sum_{i:z_i=k} (y_i - \hat{\mu}_k) \top (y_i - \hat{\mu}_k)
\]

Unsupervised classification — Unknown parameters

If $\pi_k$ and the parameters $\mu_k$ and $\Sigma_k$ of the data distribution $p(y_i | z_i = k)$ are unknown, and we do not have any training data, we have to estimate everything.

K-Means: Directly aimed toward classification. Fast and simple, but relies on model assumptions that are rarely fulfilled.

Gibbs sampling: More complicated, but can handle a broad range of problems. Estimates the model parameters.

The K-means algorithm

The K-means algorithm is a simple, popular algorithm for separating data into different clusters.

1. Select $K$ data-points at random, as initial cluster centres.
2. Assign all data points to their nearest cluster centre.
3. Compute the mean within each cluster, and let these be the new cluster centres.
4. Repeat from 2.

Drawbacks of K-means:
- Handling of different $\pi_k$?
- Different variation within clusters?

The more general statistical model can be handled by a Gaussian mixture model.
Gaussian mixture models

We can write the model as a hierarchical model, with prior class probabilities $\pi_k$ and within-class distributions $y_i | z_i = k \in N(\mu_k, \Sigma_k)$.

- This generates a Gaussian mixture model for pixel $i$, with density

$$p(y_i) = \sum_{k=1}^{K} p(y_i | z_i = k) \pi_k.$$ 

- The density for an entire image becomes

$$p(y_1, \ldots, y_n) = \prod_{i=1}^{n} \sum_{k=1}^{K} p(y_i | z_i = k) \pi_k.$$ 

- Parameter estimates can be obtained by maximising $p(y_1, \ldots, y_n)$.

Comparing the GMM and K-means

- K-means assumes equal amounts of all classes, i.e. $\pi_k = 1/K$.
- Additionally K-means implicitly "assumes" equal, isotropic variance for all classes, i.e. $\Sigma_k = \sigma^2 I$.
- K-means assigns 0/1 posterior class-probabilities, $p(z_i = k | y_i)$; this implies $\sigma^2 \rightarrow 0$. 

Comparing the GMM and K-means