Spatial Statistics with Image Analysis
Lecture 9

Johan Lindström

November 30, 2018

Lecture L09

Repetition
  Column stacking
  Vector pixels

Principal component analysis
  EIG vs SVD

Object classification
  Bayesian classification
  Supervised classification
  Unsupervised classification
  K-means algorithm

Gaussian mixture models

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.

$$Y_{\text{col-stacked}} = \begin{bmatrix}
y_{1,1,1} & \cdots & y_{1,1,d} \\
y_{2,1,1} & \cdots & y_{2,1,d} \\
\vdots & \ddots & \vdots \\
y_{m,1,1} & \cdots & y_{m,1,d} \\
y_{1,2,1} & \cdots & y_{1,2,d} \\
\vdots & \ddots & \vdots \\
y_{m,n,1} & \cdots & y_{m,n,d}
\end{bmatrix}$$
Vector pixels

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 $\Sigma = \mathcal{V}(y)$.
- The covariance can be estimated as $\hat{\Sigma} = \frac{1}{N} (Y - \overline{Y})^\top(Y - \overline{Y})$.
- 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, $\Sigma$:

\[
\Sigma = P \Lambda P^\top, \quad PP^\top = P^\top P = I,
\]

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

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

\[
C(y_i) = \Sigma = P \Lambda P^\top
\]

\[
C(y,P) = P^\top C(y_i) P = P^\top P \Lambda P^\top P = \Lambda
\]

- 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. $\text{Sigma}=(y' \ast y)/\text{size}(y,1)$, compute the eigenvectors and eigenvalues of $\Sigma$ as $[P,\Lambda]=\text{eig}(\text{Sigma})$, and sort the eigenvectors in decreasing eigenvalue order.

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

This option usually is numerical more stable.

### Singular Value Decomposition

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

$$\frac{y}{\sqrt{N}} = USV^T, \quad U^TU = I, \quad S = \text{diag}(s_1, s_2, \ldots, s_d), \quad V^TV = I.$$ 

This option usually is numerical more stable.
Singular Value Decomposition

For the singular value decomposition of $y$ we have that

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

Identifying parts gives:

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

$$p(z_i = k|y_i) = \frac{p(y_i|z_i = k)p(z_i = k)}{p(y_i)} = \frac{p(y_i|z_i = k)\pi_k}{\sum_l p(y_i|z_i = l)\pi_l}$$

- 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.
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.

\[
\begin{align*}
    n_k &= \sum_{i: z_i = k} 1 \\
    \bar{n}_k &= \frac{n_k}{\sum_{k=1}^{K} n_k} \\
    \hat{\mu}_k &= \frac{1}{n_k} \sum_{i: z_i = k} y_i \\
    \hat{\Sigma}_k &= \frac{1}{n_k - d} \sum_{i: z_i = k} (y_i - \hat{\mu}_k)^\top (y_i - \hat{\mu}_k)
\end{align*}
\]

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 $n_k$?
- 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 $\mathbf{y}_i | z_i = k \in \mathcal{N} (\mu_k, \Sigma_k)$.

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

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

- The density for an entire image becomes

$$p(\mathbf{y}_1, \ldots, \mathbf{y}_n) = \prod_{i=1}^{n} \sum_{k=1}^{K} p(\mathbf{y}_i | z_i = k) \pi_k.$$ 

- Parameter estimates can be obtained by maximising $p(\mathbf{y}_1, \ldots, \mathbf{y}_n)$.

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, $p(z_i = k | \mathbf{y}_i)$; this implies $\sigma^2 \to 0$. 

![Diagram](attachment:image.png)