Machine Learning - Lecture 3

Mixture Models and EM

27.04.2010

Bastian Leibe
RWTH Aachen
http://www.mmp.rwth-aachen.de/
leibe@umic.rwth-aachen.de

Many slides adapted from B. Schiele
Announcements

• Exercise 1 due tonight
  - Bayes decision theory
  - Maximum Likelihood
  - Kernel density estimation / k-NN
  ⇒ Submit your results to Georgios until this evening.

• Exercise modalities
  - Need to reach ≥ 50% of the points to qualify for the exam!
  - You can work in teams of up to 2 people.
  - If you work in a team
    - Turn in a single solution
    - But put both names on it
Course Outline

- **Fundamentals (2 weeks)**
  - Bayes Decision Theory
  - Probability Density Estimation

- **Discriminative Approaches (4 weeks)**
  - Linear Discriminant Functions
  - Support Vector Machines
  - Ensemble Methods & Boosting
  - Randomized Trees, Forests & Ferns

- **Generative Models (4 weeks)**
  - Bayesian Networks
  - Markov Random Fields

- **Unifying Perspective (2 weeks)**
Recap: Gaussian (or Normal) Distribution

- One-dimensional case
  - Mean $\mu$
  - Variance $\sigma^2$

  \[ \mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma}} \exp \left\{ -\frac{(x - \mu)^2}{2\sigma^2} \right\} \]

- Multi-dimensional case
  - Mean $\mu$
  - Covariance $\Sigma$

  \[ \mathcal{N}(x|\mu, \Sigma) = \frac{1}{(2\pi)^{D/2} |\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right\} \]

Image source: C.M. Bishop, 2006
Recap: Maximum Likelihood Approach

- Computation of the likelihood
  - Single data point: \( p(x_n|\theta) \)
  - Assumption: all data points \( X = \{x_1, \ldots, x_n\} \) are independent
    \[
    L(\theta) = p(X|\theta) = \prod_{n=1}^{N} p(x_n|\theta)
    \]
  - Log-likelihood
    \[
    E(\theta) = - \ln L(\theta) = - \sum_{n=1}^{N} \ln p(x_n|\theta)
    \]

- Estimation of the parameters \( \theta \) (Learning)
  - Maximize the likelihood (=minimize the negative log-likelihood)
    \Rightarrow Take the derivative and set it to zero.
    \[
    \frac{\partial}{\partial \theta} E(\theta) = - \sum_{n=1}^{N} \frac{\partial}{\partial \theta} p(x_n|\theta) \frac{1}{p(x_n|\theta)} = 0
    \]
Recap: Bayesian Learning Approach

- Bayesian view:
  - Consider the parameter vector $\theta$ as a random variable.
  - When estimating the parameters, what we compute is
    \[
    p(x|X) = \int p(x, \theta|X) d\theta
    \]
    \[
    p(x, \theta|X) = p(x|\theta, X)p(\theta|X)
    \]
    \[
    p(x|X) = \int p(x|\theta)p(\theta|X) d\theta
    \]
    Assumption: given $\theta$, this doesn’t depend on $X$ anymore
    This is entirely determined by the parameter $\theta$ (i.e. by the parametric form of the pdf).
Recap: Bayesian Learning Approach

• Discussion

Likelihood of the parametric form $\theta$ given the data set $X$.

Estimate for $x$ based on parametric form $\theta$

Prior for the parameters $\theta$

$$p(x|X) = \frac{\int p(x|\theta) L(\theta) p(\theta) \, d\theta}{\int L(\theta) p(\theta) \, d\theta}$$

Normalization: integrate over all possible values of $\theta$

- The more uncertain we are about $\theta$, the more we average over all possible parameter values.
Recap: Histograms

- Basic idea:
  - Partition the data space into distinct bins with widths $\Delta_i$ and count the number of observations, $n_i$, in each bin.
  
  $$p_i = \frac{n_i}{N \Delta_i}$$

  - Often, the same width is used for all bins, $\Delta_i = \Delta$.

  - This can be done, in principle, for any dimensionality $D$...

  ...but the required number of bins grows exponentially with $D$!

Image source: C.M. Bishop, 2006
Recap: Kernel Methods

\[ p(x) \approx \frac{K}{NV} \]

- **Kernel methods**
  - Place a *kernel window* \( k \) at location \( x \) and count how many data points fall inside it.

- **K-Nearest Neighbor**
  - Increase the volume \( V \) until the \( K \) next data points are found.

This slide is adapted from Bernt Schiele.
Topics of This Lecture

- **Mixture distributions**
  - Mixture of Gaussians (MoG)
  - Maximum Likelihood estimation attempt

- **K-Means Clustering**
  - Algorithm
  - Applications

- **EM Algorithm**
  - Credit assignment problem
  - MoG estimation
  - EM Algorithm
  - Interpretation of K-Means
  - Technical advice

- **Applications**
Mixture Distributions

- A single parametric distribution is often not sufficient
  - E.g. for multimodal data

![Single Gaussian vs Mixture of two Gaussians](image-source: C.M. Bishop, 2006)
Mixture of Gaussians (MoG)

- Sum of $M$ individual Normal distributions

In the limit, every smooth distribution can be approximated this way (if $M$ is large enough)

$$p(x|\theta) = \sum_{j=1}^{M} p(x|\theta_j)p(j)$$
Mixture of Gaussians

\[ p(x|\theta) = \sum_{j=1}^{M} p(x|\theta_j)p(j) \]

\[ p(x|\theta_j) = \mathcal{N}(x|\mu_j, \sigma_j^2) = \frac{1}{\sqrt{2\pi}\sigma_j} \exp \left\{ -\frac{(x - \mu_j)^2}{2\sigma_j^2} \right\} \]

\[ p(j) = \pi_j \text{ with } 0 \leq \pi_j \leq 1 \text{ and } \sum_{j=1}^{M} \pi_j = 1. \]

- Notes
  - The mixture density integrates to 1:
    \[ \int p(x)dx = 1 \]
  - The mixture parameters are
    \[ \theta = (\pi_1, \mu_1, \sigma_1, \ldots, \pi_M, \mu_M, \sigma_M) \]

Likelihood of measurement \( x \) given mixture component \( j \)

Prior of component \( j \)
Mixture of Gaussians (MoG)

• “Generative model”

\[
p(j) = \pi_j
\]

\[
p(x) = \sum_{j=1}^{M} p(x|\theta_j)p(j)
\]

Slide credit: Bernt Schiele
Mixture of Multivariate Gaussians
Mixture of Multivariate Gaussians

- **Multivariate Gaussians**

\[
p(x|\theta) = \sum_{j=1}^{M} p(x|\theta_j)p(j)
\]

\[
p(x|\theta_j) = \frac{1}{(2\pi)^{D/2}|\Sigma_j|^{1/2}} \exp \left\{ -\frac{1}{2}(x - \mu_j)^T \Sigma_j^{-1} (x - \mu_j) \right\}
\]

- **Mixture weights / mixture coefficients:**

\[p(j) = \pi_j \text{ with } 0 \leq \pi_j \leq 1 \text{ and } \sum_{j=1}^{M} \pi_j = 1\]

- **Parameters:**

\[
\theta = (\pi_1, \mu_1, \Sigma_1, \ldots, \pi_M, \mu_M, \Sigma_M)
\]
Mixture of Multivariate Gaussians

- “Generative model”

\[ p(j) = \pi_j \]

\[ p(x|\theta) = \sum_{j=1}^{3} \pi_j p(x|\theta_j) \]
Mixture of Gaussians - 1st Estimation Attempt

- **Maximum Likelihood**
  
  - Minimize $E = -\ln L(\theta) = -\sum_{n=1}^{N} \ln p(x_n|\theta)$
  
  - Let’s first look at $\mu_j$:
    
    $$\frac{\partial E}{\partial \mu_j} = 0$$

  - We can already see that this will be difficult, since
    
    $$\ln p(X|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k) \right\}$$

    This will cause problems!
Mixture of Gaussians - 1\textsuperscript{st} Estimation Attempt

\textbf{Minimization:}

\[
\frac{\partial E}{\partial \mu_j} = - \sum_{n=1}^{N} \frac{\partial}{\partial \mu_j} p(x_n | \theta_j) \frac{\sum_{k=1}^{K} p(x_n | \theta_k)}{\sum_{k=1}^{K} p(x_n | \theta_k)}
\]

\[
= - \sum_{n=1}^{N} \left( \Sigma^{-1}(x_n - \mu_j) \frac{p(x_n | \theta_j)}{\sum_{k=1}^{K} p(x_n | \theta_k)} \right)
\]

\[
= - \Sigma^{-1} \sum_{n=1}^{N} (x_n - \mu_j) \frac{\pi_j \mathcal{N}(x_n | \mu_j, \Sigma_j)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k)} = 0
\]

\textbf{We thus obtain}

\[
\Rightarrow \mu_j = \frac{\sum_{n=1}^{N} \gamma_j(x_n) x_n}{\sum_{n=1}^{N} \gamma_j(x_n)}
\]

“responsibility” of component \(j\) for \(x_n\)

B. Leibe
Mixture of Gaussians - 1st Estimation Attempt

- But...

\[
\mu_j = \frac{\sum_{n=1}^{N} \gamma_j(x_n)x_n}{\sum_{n=1}^{N} \gamma_j(x_n)} \quad \gamma_j(x_n) = \frac{\pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)}{\sum_{k=1}^{N} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}
\]

- I.e. there is no direct analytical solution!

\[
\frac{\partial E}{\partial \mu_j} = f(\pi_1, \mu_1, \Sigma_1, \ldots, \pi_M, \mu_M, \Sigma_M)
\]

- Complex gradient function (non-linear mutual dependencies)
- Optimization of one Gaussian depends on all other Gaussians!
- It is possible to apply iterative numerical optimization here, but in the following, we will see a simpler method.
Mixture of Gaussians - Other Strategy

- Other strategy:

- Observed data: 
- Unobserved data: 
  - Unobserved = “hidden variable”: \( j | x \)
  
\[
\begin{align*}
h(j = 1 | x_n) &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
h(j = 2 | x_n) &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}
\end{align*}
\]
Mixture of Gaussians - Other Strategy

- Assuming we knew the values of the hidden variable...

\[
\begin{align*}
\mu_1 &= \frac{\sum_{n=1}^{N} h(j = 1|x_n)x_n}{\sum_{i=1}^{N} h(j = 1|x_n)} \\
\mu_2 &= \frac{\sum_{n=1}^{N} h(j = 2|x_n)x_n}{\sum_{i=1}^{N} h(j = 2|x_n)}
\end{align*}
\]

ML for Gaussian #1

\[
h(j = 1|x_n) = \begin{cases} 
1 & j = 1 \\
0 & j \neq 1 
\end{cases}
\]

ML for Gaussian #2

\[
h(j = 2|x_n) = \begin{cases} 
0 & j = 2 \\
1 & j \neq 2 
\end{cases}
\]
Mixture of Gaussians - Other Strategy

- Assuming we knew the mixture components...

- Bayes decision rule: Decide $j = 1$ if

$$p(j = 1 | x_n) > p(j = 2 | x_n)$$
Mixture of Gaussians - Other Strategy

- Chicken and egg problem - what comes first?

We don’t know any of those!

- In order to break the loop, we need an estimate for $j$.
  - E.g. by clustering...
Clustering with Hard Assignments

• Let’s first look at clustering with “hard assignments”
Topics of This Lecture

- Mixture distributions
  - Mixture of Gaussians (MoG)
  - Maximum Likelihood estimation attempt

- K-Means Clustering
  - Algorithm
  - Applications

- EM Algorithm
  - Credit assignment problem
  - MoG estimation

- Applications

B. Leibe
K-Means Clustering

- Iterative procedure
  1. Initialization: pick $K$ arbitrary centroids (cluster means)
  2. Assign each sample to the closest centroid.
  3. Adjust the centroids to be the means of the samples assigned to them.
  4. Go to step 2 (until no change)

- Algorithm is guaranteed to converge after finite #iterations.
  - Local optimum
  - Final result depends on initialization.

Slide credit: Bernt Schiele
K-Means - Example with K=2

Image source: C.M. Bishop, 2006
K-Means Clustering

- K-Means optimizes the following objective function:

\[ J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \| x_n - \mu_k \|^2 \]

where

\[ r_{nk} = \begin{cases} 
1 & \text{if } k = \text{arg min}_j \| x_n - \mu_j \|^2 \\
0 & \text{otherwise.} 
\end{cases} \]

- In practice, this procedure usually converges quickly to a local optimum.
Example Application: Image Compression

- Take each pixel as one data point.
- Set the pixel color to the cluster mean.

K-Means Clustering

Image source: C.M. Bishop, 2006
Example Application: Image Compression

$K = 2$  $K = 3$  $K = 10$  Original image

Image source: C.M. Bishop, 2006
Summary K-Means

- **Pros**
  - Simple, fast to compute
  - Converges to local minimum of within-cluster squared error

- **Problem cases**
  - Setting k?
  - Sensitive to initial centers
  - Sensitive to outliers
  - Detects spherical clusters only

- **Extensions**
  - Speed-ups possible through efficient search structures
  - General distance measures: k-medoids

Slide credit: Kristen Grauman
Topics of This Lecture

- Mixture distributions
  - Mixture of Gaussians (MoG)
  - Maximum Likelihood estimation attempt

- K-Means Clustering
  - Algorithm
  - Applications

- EM Algorithm
  - Credit assignment problem
  - MoG estimation
  - EM Algorithm
  - Interpretation of K-Means
  - Technical advice

- Applications
EM Clustering

- Clustering with “soft assignments”
  - Expectation step of the EM algorithm

\[ f(x) \]

\[
p(j \mid x) = \begin{cases} 
0.99 & \text{if } j = 1 \\
0.01 & \text{if } j = 2 
\end{cases}
\]

\[
p(1 \mid x) = 0.99 \quad 0.8 \quad 0.2 \quad 0.01
\]

\[
p(2 \mid x) = 0.01 \quad 0.2 \quad 0.8 \quad 0.99
\]
EM Clustering

- Clustering with “soft assignments”
  - Maximization step of the EM algorithm

\[
\mu_j = \frac{\sum_{n=1}^{N} p(j|x_n)x_n}{\sum_{n=1}^{N} p(j|x_n)}
\]

\[
p(1|x) = \begin{pmatrix} 0.99 & 0.8 & 0.2 & 0.01 \end{pmatrix}
\]

\[
p(2|x) = \begin{pmatrix} 0.01 & 0.2 & 0.8 & 0.99 \end{pmatrix}
\]

Maximum Likelihood estimate
Credit Assignment Problem

"Credit Assignment Problem"

- If we are just given $x$, we don’t know which mixture component this example came from

$$p(x|\theta) = \sum_{j=1}^{2} \pi_j p(x|\theta_j)$$

- We can however evaluate the posterior probability that an observed $x$ was generated from the first mixture component.

$$p(j = 1|x, \theta) = \frac{p(j = 1, x|\theta)}{p(x|\theta)}$$

$$p(j = 1, x|\theta) = p(x|j = 1, \theta)p(j = 1) = p(x|\theta_1)p(j = 1)$$

$$p(j = 1|x, \theta) = \frac{p(x|\theta_1)p(j = 1)}{\sum_{j=1}^{2} p(x|\theta_j)p(j)}$$
Mixture Density Estimation Example

• Example
  
  ➢ Assume we want to estimate a 2-component MoG model
    \[
    p(x|\theta) = \sum_{j=1}^{2} \pi_j p(x|\theta_j)
    \]
    \[
    = \pi_1 p(x|\mu_1, \Sigma_1) + \pi_2 p(x|\mu_2, \Sigma_2)
    \]
  
  ➢ If each sample in the training set were labeled \( j \in \{1,2\} \) according to which mixture component (1 or 2) had generated it, then the estimation would be easy.
  
  ➢ Labeled examples
    = no credit assignment problem.
Mixture Density Estimation Example

- When examples are labeled, we can estimate the Gaussians independently
  - Using Maximum Likelihood estimation for single Gaussians.

- Notation
  - Let $l_i$ be the label for sample $x_i$
  - Let $N$ be the number of samples
  - Let $N_j$ be the number of samples labeled $j$
  - Then for each $j \in \{1,2\}$ we set
    \[
    \hat{\pi}_j \leftarrow \frac{N_j}{N} \\
    \hat{\mu}_j \leftarrow \frac{1}{N_j} \sum_{i: l_i = j} x_i \\
    \hat{\Sigma}_j \leftarrow \frac{1}{N_j} \sum_{i: l_i = j} (x_i - \hat{\mu}_j)(x_i - \hat{\mu}_j)^T
    \]
Mixture Density Estimation Example

• Of course, we don’t have such labels $l_i$...
  
  ➢ But we can guess what the labels might be based on our current mixture distribution estimate (credit assignment problem).
  
  ➢ We get soft labels or posterior probabilities of which Gaussian generated which example:

$$\gamma_j(x_i) = p(l_i = j | x_i, \theta) \sum_{j=1}^{2} \gamma_j(x_i) = 1 \quad \forall i = 1, \ldots, N$$

➤ When the Gaussians are almost identical (as in the figure), then $\gamma_1(x_i) \approx \gamma_2(x_i)$ for almost any given sample $x_i$.

⇒ Even small differences can help to determine how to update the Gaussians.
EM Algorithm

- **Expectation-Maximization (EM) Algorithm**
  - **E-Step**: softly assign samples to mixture components
    \[
    \gamma_j(x_n) \leftarrow \frac{\pi_j \mathcal{N}(x_n | \mu_j, \Sigma_j)}{\sum_{k=1}^{N} \pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k)} \quad \forall j = 1, \ldots, K, \ n = 1, \ldots, N
    \]
  - **M-Step**: re-estimate the parameters (separately for each mixture component) based on the soft assignments
    \[
    \hat{N}_j \leftarrow \sum_{n=1}^{N} \gamma_j(x_n) = \text{soft number of samples labeled } j
    \]
    \[
    \hat{N}_j^{\text{new}} \leftarrow \frac{\hat{N}_j}{N}
    \]
    \[
    \hat{\mu}_j^{\text{new}} \leftarrow \frac{1}{\hat{N}_j} \sum_{n=1}^{N} \gamma_j(x_n)x_n
    \]
    \[
    \hat{\Sigma}_j^{\text{new}} \leftarrow \frac{1}{\hat{N}_j} \sum_{n=1}^{N} \gamma_j(x_n)(x_n - \hat{\mu}_j^{\text{new}})(x_n - \hat{\mu}_j^{\text{new}})^T
    \]

Slide adapted from Bernt Schiele
EM Algorithm - An Example
EM - Technical Advice

- When implementing EM, we need to take care to avoid singularities in the estimation!
  - Mixture components may collapse on single data points.
  - E.g. consider the case $\sum_k = \sigma_k^2 I$ (this also holds in general)
  - Assume component $j$ is exactly centered on data point $x_n$. This data point will then contribute a term in the likelihood function
    \[
    \mathcal{N}(x_n|x_n, \sigma_j^2 I) = \frac{1}{\sqrt{2\pi \sigma_j}}
    \]
    - For $\sigma_j \to 0$, this term goes to infinity!
  \[\Rightarrow\] Need to introduce regularization
    - Enforce minimum width for the Gaussians

Image source: C.M. Bishop, 2006
EM - Technical Advice (2)

- EM is very sensitive to the initialization
  - Will converge to a local optimum of $E$.
  - Convergence is relatively slow.

⇒ Initialize with k-Means to get better results!
  - k-Means is itself initialized randomly, will also only find a local optimum.
  - But convergence is much faster.

- Typical procedure
  - Run k-Means $M$ times (e.g. $M = 10-100$).
  - Pick the best result (lowest error $J$).
  - Use this result to initialize EM
    - Set $\mu_j$ to the corresponding cluster mean from k-Means.
    - Initialize $\Sigma_j$ to the sample covariance of the associated data points.
K-Means Clustering Revisited

- Interpreting the procedure
  1. Initialization: pick $K$ arbitrary centroids (cluster means)
  2. Assign each sample to the closest centroid.  (E-Step)
  3. Adjust the centroids to be the means of the samples assigned to them.  (M-Step)
  4. Go to step 2 (until no change)
K-Means Clustering Revisited

• K-Means clustering essentially corresponds to a Gaussian Mixture Model (MoG or GMM) estimation with EM whenever
  - The covariances are of the $K$ Gaussians are set to $\Sigma_j = \sigma^2 I$
  - For some small, fixed $\sigma^2$

Slide credit: Bernt Schiele
Summary: Gaussian Mixture Models

- **Properties**
  - Very general, can represent any (continuous) distribution.
  - Once trained, very fast to evaluate.
  - Can be updated online.

- **Problems / Caveats**
  - Some numerical issues in the implementation
    - Need to apply regularization in order to avoid singularities.
  - EM for MoG is computationally expensive
    - Especially for high-dimensional problems!
    - More computational overhead and slower convergence than k-Means
    - Results very sensitive to initialization
    - Run k-Means for some iterations as initialization!
  - Need to select the number of mixture components $K$
    - Model selection problem (see Lecture 10)
Topics of This Lecture

- Mixture distributions
  - Mixture of Gaussians (MoG)
  - Maximum Likelihood estimation attempt

- K-Means Clustering
  - Algorithm
  - Applications

- EM Algorithm
  - Credit assignment problem
  - MoG estimation
  - EM Algorithm
  - Interpretation of K-Means
  - Technical advice

- Applications
Applications

- Mixture models are used in many practical applications.
  - Wherever distributions with complex or unknown shapes need to be represented...

- Popular application in Computer Vision
  - Model distributions of pixel colors.
  - Each pixel is one data point in e.g. RGB space.
    ⇒ Learn a MoG to represent the class-conditional densities.
    ⇒ Use the learned models to classify other pixels.

Image source: C.M. Bishop, 2006
Application: Background Model for Tracking

- **Train background MoG for each pixel**
  - Model “common“ appearance variation for each background pixel.
  - Initialization with an empty scene.
  - Update the mixtures over time
    - Adapt to lighting changes, etc.

- **Used in many vision-based tracking applications**
  - Anything that cannot be explained by the background model is labeled as foreground (=object).
  - Easy segmentation if camera is fixed.


B. Leibe

Image Source: Daniel Roth, Tobias Jägglı
Application: Image Segmentation

- User assisted image segmentation
  - User marks two regions for foreground and background.
  - Learn a MoG model for the color values in each region.
  - Use those models to classify all other pixels.

⇒ Simple segmentation procedure
  (building block for more complex applications)
Application: Color-Based Skin Detection

- Collect training samples for skin/non-skin pixels.
- Estimate MoG to represent the skin/non-skin densities.

Classify skin color pixels in novel images

References and Further Reading

• More information about EM and MoG estimation is available in Chapter 2.3.9 and the entire Chapter 9 of Bishop’s book (recommendable to read).

Christopher M. Bishop
Pattern Recognition and Machine Learning
Springer, 2006

• Additional information
  ➢ Original EM paper:
  ➢ EM tutorial:
Matlab Intro: Everything is a matrix

```
1   %% Matrix Definition 1 %%
2   A = [1 2 3 4; 5 6 7 8];
3
4   %% Matrix Definition 2 %%
5   A = [1:1:4; 5:1:8];
6
7   %% Matrix Definition 3 %%
8   for i = 1:2
9       for j = 1:4
10          A(i,j) = (i-1)*4+j;
11     end
12   end
13 |
14   %% Matrix Definition 4 %%
15   A = [];
16   A = zeros(n,n); % ones zeros eye rand randn

N = 5
v = [1 0 0]
% A scalar
v = [1; 2; 3]
% A row vector
v = v'
% A column vector
% Transpose a vector (row to column or column to row)
```
Matlab Intro: Matrix Index

```matlab
18     %% Matrix index
19     A = magic(4);
20     >> A =
21         16    2    3    13
22         5   11   10     8
23         9    7    6    12
24         4   14   15    1
25     A(1:2, 1:2)
26     >> ans =
27         16    2
28         5   11
29     A(:,1)
30     >> ans =
31         16
32         5
33         9
34         4
35     A(1,:)
36     >> ans =
37         16    2    3    13
38     A([6 7])
39     >> ans =
40         11    7
```

Tutorial adapted from W. Freeman, MIT
Matlab Intro: Manipulating Matrices

```matlab
44 A = [1 NaN 2 NaN; 3 Inf 4 Inf]; %/0=Inf 0/0=Nan
45 %% replace Nan/Inf with 0
46 for i = 1:size(A,1)
47     for j = 1:size(A,2)
48         if isnan(A(i,j)) | isinf(A(i,j))
49             A(i,j) = 0;
50         end
51     end
52 end
53
54 %% 'find'
55 a = [0 1 0 2; 0 3 0 4]
56     >> a =
57          0 1 0 2
58          0 3 0 4
59     find(a)
60         >> ans = [3 4 7 8]
61     [ii jj] = find(a);
62         >> ii = [1 2 1 2]
63         >> jj = [2 2 4 4]
64
65     nanflag = isnan(A)
66         >> nanflag =
67          0 1 0 1
68          0 0 0 0
69     infflag = isinf(A)
70         >> infflag =
71          0 0 0 0
72          0 1 0 1
```

Tutorial adapted from W. Freeman, MIT
Matlab Intro: Manipulating Matrices

```matlab
A = [1 2; 3 4];
B = [1 1; 1 -1];
A*B
>> ans =
  3 -1
  7 -1
A.*B
>> ans =
  1  2
  3 -4
A/B ≡ A * inv(B)
>> ans =
  1.5  -0.5
  3.5  -0.5
A/B
>> ans =
  1  2
  3 -4
```

```matlab
C = [A B]
>> C =
  1  2  3  4
C = [A;B]
>> C =
  1  2
  3  4
A = [1 2];
A*A' = 5
A.*A =
  1  4
sum(A.*A) = 5
```
Matlab Intro: Scripts and Functions

- Scripts are m-files containing MATLAB statements
- Functions are like any other m-file, but they accept arguments
- Name the function file the same as the function name

```matlab
myfunction.m

function y = myfunction(x)
  % Function of one argument with one return value
  a = [-2 -1 0 1]; % Have a global variable of the same name
  y = a + x;

myotherfunction.m

function [y, z] = myotherfunction(a, b)
  % Function of two arguments with two return values
  y = a + b;
  z = a - b;
```
Matlab Intro: Try to Code in Matrix Ways

```matlab
% use for loops
A = [1 2 3 4];
for i = 1:4
    A(i) = A(i)^2;
end
% or use matrix
A = [1 2 3 4];
A = A.^2;

% use for loops
A = [1 2 3 4; 5 6 7 8];
sum(A)
>> ans =
  6  8 10 12
ASUM = sum(A,2)
>> ASUM =
 10
26
for i = 1:size(A,1)
    for j = 1:size(A,2)
        APROB(i,j) = A(i,j)/ASUM(i);
    end
end

% use matrix
A = [1 2 3 4; 5 6 7 8];
ASUM = sum(A,2)
APROB = A./repmat(ASUM, 1, size(A,2));
repmat(ASUM, 1, size(A,2))
>> ans =
 10 10 10 10
26 26 26 26
```

B. Leibe
Matlab Intro: Important Commands

- `whos` – List variables in workspace
- `help` – Get help for any command
- `lookfor` – Search for keywords
- `clear/clear x` – Erase a variable/all variables
- `save` – Save the workspace
- `load` – Load a saved workspace
- `keyboard` – Enter debugging mode (until `dbquit`)
function out = webcam
% uses "Image Acquisition Toolbox"
adaptorName = 'winvideo';
vidFormat = 'I420_320x240';
vidObj1= videoinput(adaptorName, 1, vidFormat);
set(vidObj1, 'ReturnedColorSpace', 'rgb');
set(vidObj1, 'FramesPerTrigger', 1);
out = vidObj1 ;

cam = webcam();
img=getsnapshot(cam);