Supervised Learning Evaluation, Computer Vision

Artificial Intelligence @ Allegheny College

Janyl Jumadinova

February 3, 2020
A feature is a measurable property or a characteristic of the object we are trying to analyze (columns in a data set).
A **feature** is a measurable property or a characteristic of the object we are trying to analyze (columns in a data set).

**Discrimination** attempts to separate distinct sets of objects.

**Classification** attempts to allocate new objects to predefined groups.
Cost ratio is a ratio of false positives (given condition is present when it is not) to false negatives (given condition is not present when it actually is).

Confusion matrix (error matrix): a table to visualize the performance of an algorithm with rows/columns representing instances of predictions and columns/rows representing instances of actual class.
Confusion Matrix

<table>
<thead>
<tr>
<th>True 1</th>
<th>Predicted 1</th>
<th>Predicted 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>d</td>
</tr>
</tbody>
</table>

threshold

accuracy = (a+d) / (a+b+c+d)
Confusion Matrix

- **a** is a true positive (TP).
- **d** is a true negative (FN).
- **c** is a false positive (FP).
- **b** is a false negative (FN).

\[
\text{accuracy} = \frac{a+d}{a+b+c+d}
\]
Classification Accuracy

Number of correctly classified examples divided by the total number of examples.

\[
Accuracy = \frac{TP + TN}{TP + TN + FP + FN}
\]  

(1)
Classification Accuracy

Number of correctly classified examples divided by the total number of examples.

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}
\]

\[
\text{Error} = 1 - \text{Accuracy} \tag{2}
\]
Performance Measures

\[
Recall = \frac{TP}{TP + FN} \quad (3)
\]

Higher the recall the better class is correctly recognized (small number of FN).

Higher the precision the better indication of an example labeled as positive being indeed positive (small number of FP).

High recall, low precision: Most of the positive examples are correctly recognized (low FN) but there are a lot of false positives.

Low recall, high precision: Miss a lot of positive examples (high FN) but those we predict as positive are indeed positive (low FP).

\[
F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (5)
\]

F1 Score is used to find a balance between Precision and Recall.
Performance Measures

\[
Recall = \frac{TP}{TP + FN} \quad (3)
\]

Higher the recall the better class is correctly recognized (small number of FN).

\[
Precision = \frac{TP}{TP + FP} \quad (4)
\]

Higher the precision the better indication of an example labeled as positive being indeed positive (small number of FP).

\[
F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \quad (5)
\]

F1 Score is used to find a balance between Precision and Recall.
Performance Measures

Recall = \frac{TP}{TP + FN} \quad (3)

Higher the recall the better class is correctly recognized (small number of FN).

Precision = \frac{TP}{TP + FP} \quad (4)

Higher the precision the better indication of an example labeled as positive being indeed positive (small number of FP).

- High recall, low precision: Most of the positive examples are correctly recognized (low FN) but there are a lot of false positives.
- Low recall, high precision: Miss a lot of positive examples (high FN) but those we predict as positive are indeed positive (low FP).
Performance Measures

Recall = \frac{TP}{TP + FN} \quad (3)

Higher the recall the better class is correctly recognized (small number of FN).

Precision = \frac{TP}{TP + FP} \quad (4)

Higher the precision the better indication of an example labeled as positive being indeed positive (small number of FP).

- High recall, low precision: Most of the positive examples are correctly recognized (low FN) but there are a lot of false positives.
- Low recall, high precision: Miss a lot of positive examples (high FN) but those we predict as positive are indeed positive (low FP).

F1 = 2 \frac{Precision \times Recall}{Precision + Recall} \quad (5)

F1 Score is used to find a balance between Precision and Recall.
Performance Measures

- Receiver Operator Characteristic **ROC curve**: plot of TP vs. FP.
Performance Measures

- Receiver Operator Characteristic **ROC curve**: plot of TP vs. FP.
- ROC Curves summarize the trade-off between the true positive rate and false positive rate for a predictive model using different probability thresholds.

ROC curves are appropriate when the observations are balanced between each class, whereas precision-recall curves are appropriate for imbalanced datasets.
Receiver Operator Characteristic **ROC curve**: plot of TP vs. FP.

**ROC Curves** summarize the trade-off between the true positive rate and false positive rate for a predictive model using different probability thresholds.

**Precision-Recall curves** summarize the trade-off between the true positive rate and the positive predictive value for a predictive model using different probability thresholds.
Performance Measures

- Receiver Operator Characteristic **ROC curve**: plot of TP vs. FP.
- ROC Curves summarize the trade-off between the true positive rate and false positive rate for a predictive model using different probability thresholds.
- Precision-Recall curves summarize the trade-off between the true positive rate and the positive predictive value for a predictive model using different probability thresholds.
- ROC curves are appropriate when the observations are balanced between each class, whereas precision-recall curves are appropriate for imbalanced datasets.
What-If Tool

Smile Detection Demo

https://pair-code.github.io/what-if-tool/
Computer Vision

Make computers understand images and video.
Computer Vision

Make computers understand images and video.
Computer Vision

Make computers understand images and video
Computer Vision

Make computers understand images and video
Computer Vision

- What kind of scene?
- Where are the cars?
- How far is the building?
Why computer vision matters?

Safety

Health

Security

Comfort

Fun

Access
Applications of Computer Vision

“Face Recognition”  “Pose Estimation”  “Body Tracking”

“Speech Reading”  “Palm Recognition”  “Car Tracking”
Segmentation

- Compact representation for image data in terms of a set of components.
Segmentation

- Compact representation for image data in terms of a set of components.
- Components share “common” visual properties.
Segmentation

- Compact representation for image data in terms of a set of components.
- Components share “common” visual properties.
- Properties can be defined at different level of abstractions.
Segmentation

- Compact representation for image data in terms of a set of components.
- Components share “common” visual properties.
- Properties can be defined at different level of abstractions.

From: https://docs.opencv.org
Segmentation

- **Tokens**
  - whatever we need to group (pixels, points, surface elements, etc.).
Segmentation

- **Tokens**
  - whatever we need to group (pixels, points, surface elements, etc.).

- **Bottom up segmentation**
  - tokens belong together because they are locally coherent.

From: “Combining Top-Down and Bottom-Up Segmentation” by Eran Borenstein, Eitan Sharon, Shimon Ullman
Segmentation

- **Tokens**
  - whatever we need to group (pixels, points, surface elements, etc.).

- **Bottom up segmentation**
  - tokens belong together because they are locally coherent.

- **Top down segmentation**
  - tokens belong together because they lie on the same object.
Segmentation

- **Tokens**
  - whatever we need to group (pixels, points, surface elements, etc.).

- **Bottom up segmentation**
  - tokens belong together because they are locally coherent.

- **Top down segmentation**
  - tokens belong together because they lie on the same object.

![Segmentation Diagram](image)

**Figure 1**: The relative merits of the bottom-up and the top-down.
Clustering image elements that “belong together”

- **Partitioning**
  - Divide into regions/sequences with coherent internal properties.

- **Grouping**
  - Identify sets of coherent tokens in image.
An open source BSD licensed computer vision library
- Patent-encumbered code isolated into “non-free” module (SIFT, SURF, some of the Face Detectors, etc.)
Available on all major platforms
- Android, iOS, Linux, Mac OS X, Windows
Written primarily in C++
- Bindings available for Python, Java, even MATLAB (in 3.0).
Well documented at http://docs.opencv.org
Source available at https://github.com/Itseez/opencv
OpenCV

Image Processing
- Filters
- Transformations
- Edges, contours
- Robust features
- Segmentation

Video, Stereo, 3D
- Calibration
- Pose estimation
- Optical Flow
- Detection and recognition
- Depth
Load an image from the disk, display it on our screen, and write it to file in a different format.
OpenCV: Pixel

- **Grayscale**: each pixel has a value between 0 (black) and 255 (white)
  - values between 0 and 255 are varying shades of gray.
Grayscale: each pixel has a value between 0 (black) and 255 (white)
- values between 0 and 255 are varying shades of gray.

Color: pixels are normally represented in the RGB color space
- one value for the Red component, one for Green, and one for Blue,
- each of the three colors is represented by an integer in the range 0 to 255,
- how “much” of the color there is.
OpenCV: Coordinate System

- The point (0, 0) corresponds to the upper left corner of the image
- x value increases as we move to the right
- y value increases as we move down
OpenCV: Image Representation

- OpenCV represents images as NumPy arrays (matrices).
- NumPy is a library for the Python programming language that provides support for large, multi-dimensional arrays.
- To access a pixel value, we need to supply the x and y coordinates of the pixel.
- OpenCV actually stores RGB values in the order of Blue, Green, and Red.
How to input or output an image?
How to input or output an image?

**Load Image**
Read image from disk.
```cpp
cv::imread(filename, 0/1);
```
0: read as grayscale image
1: read as color image

**Save Image**
Write image to disk.
```cpp
cv::imwrite(filename, im);
```

**Visualize Image**
Show image in a window.
```cpp
cv::imshow(title, im);
```
*Note: if CV_32FC1, the gray value range is 0 to 1. Everything above 1 is white and everything below 0 is black.*

**Waitkey**
Waits n milliseconds for user input.
```cpp
cv::waitkey(n);
```
If n == -1, it waits forever.
*Note: There must be a waitkey to show the image.*
Drawing Primitives

\[
\text{cv::line}(\text{im}, p_1, p_2, \text{color}, \text{thickness})
\]

\[
\text{cv::Point}(x, y)
\]

\[
\text{cv::circle}(\text{im}, c, r, \text{color}, \text{thickness})
\]

\[
\text{CV_RGB}(r, g, b)
\]
rectangle = np.zeros((300, 300), dtype = "uint8")
cv2.rectangle(rectangle, (25, 25), (275, 275), 255, -1)
Examine every pixel in the input images:

- `cv2.bitwise_and` (used in masking example): if both pixels have a value $> 0$, the output pixel is set to 255 in the output image, otherwise it is 0.

- `cv2.bitwise_or`: if either of the pixels have a value $> 0$, the output pixel is set to 255 in the output image, otherwise it is 0.

- `cv2.bitwise_xor`: same as OR, with a restriction: both pixels are not allowed to have values $> 0$.

- `cv2.bitwise_not`: pixels with a value of 255 become 0, pixels with a value of 0 become 255.
1. Load an image from the disk, display it on our screen, and write it to file in a different format.

2. Access and manipulate pixels.