Project 2 Note
Deliverable 1
Project 2: Dataset

• The MSR Daily Activity 3D Dataset

• The dataset is divided into Training and Testing sets

• We only use six activity categories.

- CheerUp (a08)
- TossPaper (a10)
- LieOnSofa (a12)
- Walking (a13)
- StandUp (a15)
- SitDown (a16)
Project 2: Dataset

• Project Datasets

48 Lines

72 Lines
Project 2: Dataset

• Project Datasets

48 Lines

72 Lines

Output
Project 2: Dataset

- Datasets

128 Lines

192 Lines
Project 2: Dataset

• Data format

When you open a data instance file, say `a12_s08_e02_skeleton_proj.txt`, you will see something like:

```
1 1 0.326 -0.101  2.111
1 2 0.325 -0.058  2.152
1 3 0.319  0.194  2.166
```

Each row contains five values, representing:
1) `frame_id`,
2) `joint_id`,
3) `joint_position_x`,
4) `joint_position_y`,
5) `joint_position_z`.

These joint positions are in real-world 3D coordinates with a unit of meters. Each frame contains 20 rows that contain information of all joints in the frame. The skeleton data from all frames of an instance are included in the same data file (like `a12_s08_e02_skeleton_proj.txt`).

Fig. 2: Skeleton joint names and indices from Kinect SDK.
Relative Angles and Distances (RAD)

Algorithm 1: RAD representation using star skeletons

Input : Training set Train or testing set Test
Output : rad_dl or rad_dl.t

1: for each instance in Train or Test do
2:     for frame $t = 1, ..., T$ do
3:         Select joints that form a star skeleton (Figure 3);
4:         Compute and store distances between body
5:         extremities to body center ($d^t_1, ..., d^t_5$);
6:         Compute and store angles between two adjacent body
7:         extremities ($\theta^t_1, ..., \theta^t_5$);
8:     end
9:     Compute a histogram of $N$ bins for each $d_i = \{d^t_i\}_{t=1}^T$, $i = 1, ..., 5$;
10:    Compute a histogram of $M$ bins for each $\theta_i = \{\theta^t_i\}_{t=1}^T$, $i = 1, ..., 5$;
11:    Normalize the histograms by dividing $T$ to compensate
12:    for different number of frames in a data instance;
13:    Concatenate all normalized histograms into a
14:    one-dimensional vector of length $5(M + N)$;
15:    Convert the feature vector as a single line in the rad_dl
16:    or rad_dl.t file.
17: end
18: return rad_dl or rad.t_d2
Algorithm 1: RAD representation using star skeletons

Input : Training set Train or testing set Test
Output: rad_d1 or rad_d1.t

1: for each instance in Train or Test do
2:     for frame $t = 1, ..., T$ do
3:         Select joints that form a star skeleton (Figure 3);
4:         Compute and store distances between body
      extremities to body center ($d_i^1, ..., d_i^5$);
5:         Compute and store angles between two adjacent body
      extremities ($\theta_i^1, ..., \theta_i^5$);
6:     end
7:     Compute a histogram of $N$ bins for each $d_i = \{d_i^t\}_{t=1}^T$,
     $i = 1, ..., 5$;
8:     Compute a histogram of $M$ bins for each $\theta_i = \{\theta_i^t\}_{t=1}^T$,
     $i = 1, ..., 5$;
9:     Normalize the histograms by dividing $T$ to compensate
      for different number of frames in a data instance;
10:    Concatenate all normalized histograms into a
       one-dimensional vector of length $5(M + N)$;
11:    Convert the feature vector as a single line in the rad_d1
       or rad_d1.t file.
12: end
13: return rad_d1 or rad.t.d2
Project 2: Histogram of Joint Position Differences (HJPD)

*Histogram of Joint Position Differences*: Given the location of a joint \((x, y, z)\) and a reference joint \((x_c, y_c, z_c)\) in the world coordinate, the joint displacement is defined as

\[
(\Delta x, \Delta y, \Delta z) = (x, y, z) - (x_c, y_c, z_c)
\]

The reference joint can be the skeleton centroid or a fixed joint. For each sequence of human skeletons representing an activity, a histogram is computed for the displacement along each dimension (e.g., \(\Delta x\), \(\Delta y\) or \(\Delta z\)). Then, the computed three histograms are concatenated into a single vector as a feature.
Project 2: Histogram of Oriented Displacement (HOD)

Histogram of Oriented Displacements: (1) project 3D trajectory of of each joint onto the 2D Cartesian planes (i.e., $xy$, $yz$, and $zx$); (2) on each 2D Cartesian plane, compute the orientation of the trajectory to the reference coordinate; (3) compute a normalized histogram of all orientations (from all joints in adjacent time).
Project 2 Deliverable 1 Submission

**CSCI473**: Implement RAD and another skeleton-based representation

- A README that provides sufficient instructions needed to compile and execute your code. Your README also needs to document your implementation information, including which joints are used in the RAD representation, how the histograms are computed, and how many bins are used.
- Your code to construct the RAD and customized representations.
- The generated representation data, including `rad_d1`, `rad_d1.t`, `cust_d1`, and `cust_d1.t`. 
Project 2 Deliverable 1 Submission

**CSCI573**: Implement RAD, HJPD, and HOD (including temporal pyramids)

- A README that provides sufficient instructions needed to compile and execute your code. Your README also needs to document your implementation information, for example, including which joints are used in the RAD representation, and how the histograms are computed and how many bins are used in your HJPD and HOD representations.
- All your code to construct the RAD, HJPD, and HOD representations.
- All the generated skeleton-based representation data, including `rad_d1`, `rad_d1.t`, `hjpd_d1`, `hjpd_d1.t`, `hod_d1`, and `hod_d1.t`. 
Project 2 Deliverable 1 Submission

• Students are required to program this project using C++ or Python in a Linux system
• Students are not required to implement the project in ROS
• Your code must be able to run on Ubuntu 14.04 LTS or Ubuntu 16.04 LTS (if needed, you may use the Linux computers in the Alamode Lab at BB136.
• Submit a single tarball, named
  \[ T1\_firstname\_lastname.tar \] (or.tar.gz)
  to the Blackboard portal named P2-T1
• Students in CSCI573 will be graded more strictly on the quality of the implementation
Project 2 Note
Deliverable 2
Project 2 Deliverable 2: CSCI 473

A. Tasks to Perform

Students in CSCI 473 need to perform the following tasks to finish Deliverable 2, including:

1) **Read and understand Section IV** as well as the **Practical Guide of LIBSVM** (the theory and practical usage of LIBSVM have been discussed in the class). Go through the examples provided by the Practical Guide.

2) Convert the training and testing files (i.e., the outputs of your code to build the representations in Deliverable 1) to a format that can be used by LIBSVM. Name them as `rad_d2`, `rad_d2.t`, `cust_d2`, and `cust_d2.t`.

3) Apply LIBSVM to learn a C-SVM model with the RBF kernel from the training data, and use the learned model to predict behavior labels of the testing data, which will generate a result file, e.g., `rad_d2.t.predict`, and `cust_d2.t.predict`.

4) Evaluate its performance based upon the output results using the performance metrics of **accuracy** and **confusion matrix**.
16

1. Hyper-parameter/model selection of Example 1
2. Experimental results of Example 1

$ ./svm-scale -l -1 -u 1 -s range3 svmguide3 > svmguide3.scale
$ ./svm-scale -r range3 svmguide3.t > svmguide3.t.scale
$ ./svm-train svmguide3.scale
$ ./svm-predict svmguide3.t.scale svmguide3.scale.model svmguide3.t.predict
→ Accuracy = 12.1951%

$ python easy.py svmguide3 svmguide3.t
Scaling training data...
Cross validation...
Best c=128.0, g=0.125
Training...
Scaling testing data...
Testing...
Accuracy = 87.8049% (36/41) (classification)
Project 2 Deliverable 2: CSCI 473

B. What to Submit

For the Deliverable 2, CSCI 473 students are required to submit a **single tarball**, named *T2Firstname_Lastname.tar* (or *.tar.gz*) to the Blackboard portal named P2-T2, which must contain the following items:

- The graphs of the grid search (from grid.py) obtained in the experiments for both representations (i.e., RAD and the customized representation);
- What are the “best” values of $C$ and $\gamma$ of your C-SVMs for both representations, which need to be written in the README;
- All the converted representation files and output prediction result files;
- Any code you write in Deliverable 2 and the README (with the same requirement as in Deliverable 1).
A. Tasks to Perform

Students in CSCI 573 need to perform the following tasks to finish Deliverable 2, including:

1) Read and understand Section IV as well as the Practical Guide of LIBSVM (the theory and practical usage of LIBSVM have been discussed in the class). Go through the examples in the Practical Guide, and understand how to integrate LIBSVM into your source code.

2) Convert the training and testing files (i.e., the outputs of your code to build the representations in Deliverable 1) to a format that can be used by LIBSVM. This must be done for all three representations (i.e., RAD, HJPD, and HOD).

3) Apply LIBSVM to learn a C-SVM model with the RBF kernel from the training data, and use the learned model to predict behavior labels of the testing data, which will generate a result file for all the representations.

4) Write an integrated program that reads data from the training and testing directories, creates a given representation (specified by a command-line flag), performs robot learning, and outputs the information of accuracy and confusion matrix to the screen. Your implementation’s accuracy based on each representation must be better than 50%.

5) Analyze how the accuracy varies according to different numbers of bins.
B. What to Submit

For the Deliverable 2, CSCI 573 students are required to submit a **single tarball**, named `T2_firstname_lastname.tar` (or `.tar.gz`) to the Blackboard portal named P2-T2, which must contain the following items:

- The graphs of the grid search (from grid.py) obtained in the experiments for all the representations (i.e., RAD, HJPD, and HOD);
- What are the “best” values of $C$ and $\gamma$ of your C-SVMs for all the representations, which can be written in the README;
- All the converted representation files and output prediction result files;
- A figure showing how accuracy varies according to the number of bins;
- The code you write in Deliverable 2 and the README (with the same requirement as in Deliverable 1).
Evaluation

Confusion matrix and accuracy (Both CSCI473/573)

<table>
<thead>
<tr>
<th>Actual Activity Number</th>
<th>LIBSVM Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
</tr>
</tbody>
</table>
Evaluation

Sensitivity analysis (CSCI573)