Running a job in the MARS environment

This writeup assumes you've already set up the MARS conda environment.

Creating and running a job

You can submit jobs via either shell or the web interface, but I've only used the latter so here are instructions for that!

1. From the OnDemand dashboard, select Jobs>Job Composer.
2. Click New Job>From Default Template.
3. Give you job a name using the Job Options menu if desired.
4. Scroll down to view the Submit Script `main_job.sh` on the right, and click Open Editor to modify this script to do whatever you want.

For running MARS, replace this script with the code from `run_test.sh` in the Github repo:

```
#!/bin/bash
#
#SBATCH --time=60:00:00   # walltime
#SBATCH --ntasks=1   # number of processor cores (i.e. tasks)
#SBATCH --nodes=1   # number of nodes
#SBATCH --mem-per-cpu=192G   # memory per CPU core
#SBATCH -J "run_training"   # job name

source activate mars_tf
cd ~/MARS_train_infer_CMS/
python run_training.py sniff_face both > test_output/mars_log.txt
```

The first few lines are Slurm commands characterizing the job.

The last three lines will activate our conda environment, cd into the git repo, and call `run_training.py` with inputs `sniff_face` (the behavior to train on) and `both` (ie run both training and testing).

5. Finally, click the green Submit button. Your job's status will change from `Queued` to `Running` to eventually `Completed`.

Monitoring job status

While the job is running, output of the python script will be sent to `~/MARS_train_infer_CMS/test_output/mars_log.txt` (you may need to create the `test_output` directory.)

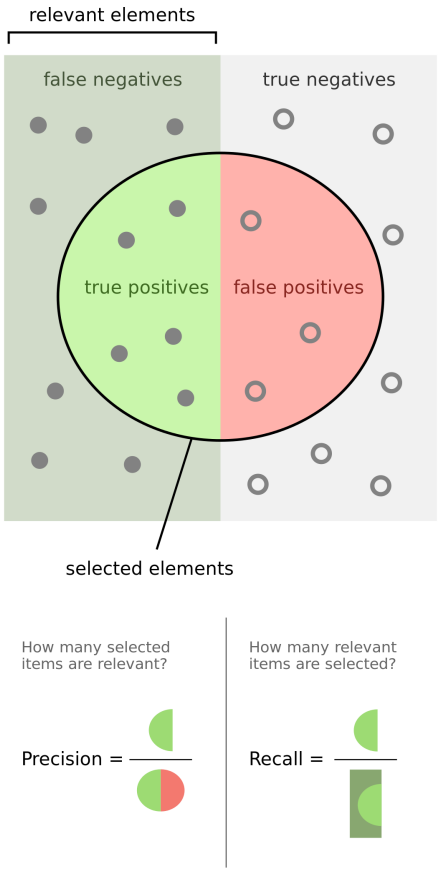
Terminal outputs, such as error messages, are sent to `slurm-XXXX.out` files in this job's folder, which will be located somewhere like `~/ondemand/data/sys/myjobs/projects/default/XX`.

In the Job Composer interface, contents of the job's folder are listed under Folder Contents: you will see `/main_job.sh` followed by one or more `slurm-XXXX.out` files. A new `*.out` file is created each time the job is submitted, and the bottom-most is the most recent (you may want to `rm *.out` periodically to make the list manageable.)

If you job completes uncharacteristically quickly, check the `*.out` file to see if any error messages arose.

Evaluating classifier performance

MARS's `run_training` script evaluates classifier performance by running the classifier on a separate test set of videos. For each tested video, MARS will report classifier performance in terms of **Precision**, **Recall**, and **F1 Score**.



Precision and Recall are bounded from 0 to 1, and are easiest to understand in terms of the diagram to the right. Essentially, *precision* reflects how many of the classifier's positive-labeled frames are actually positive, while *recall* reflects how many of the true-positive frames were actually detected.

We use them in place of accuracy because if a behavior only occurs 5% of the time, a classifier could achieve 95% *accuracy* by just reporting that the behavior never happened- whereas that classifier would have a *recall* of 0.

Lastly, the F1 score is just the harmonic mean of the precision and recall,

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$