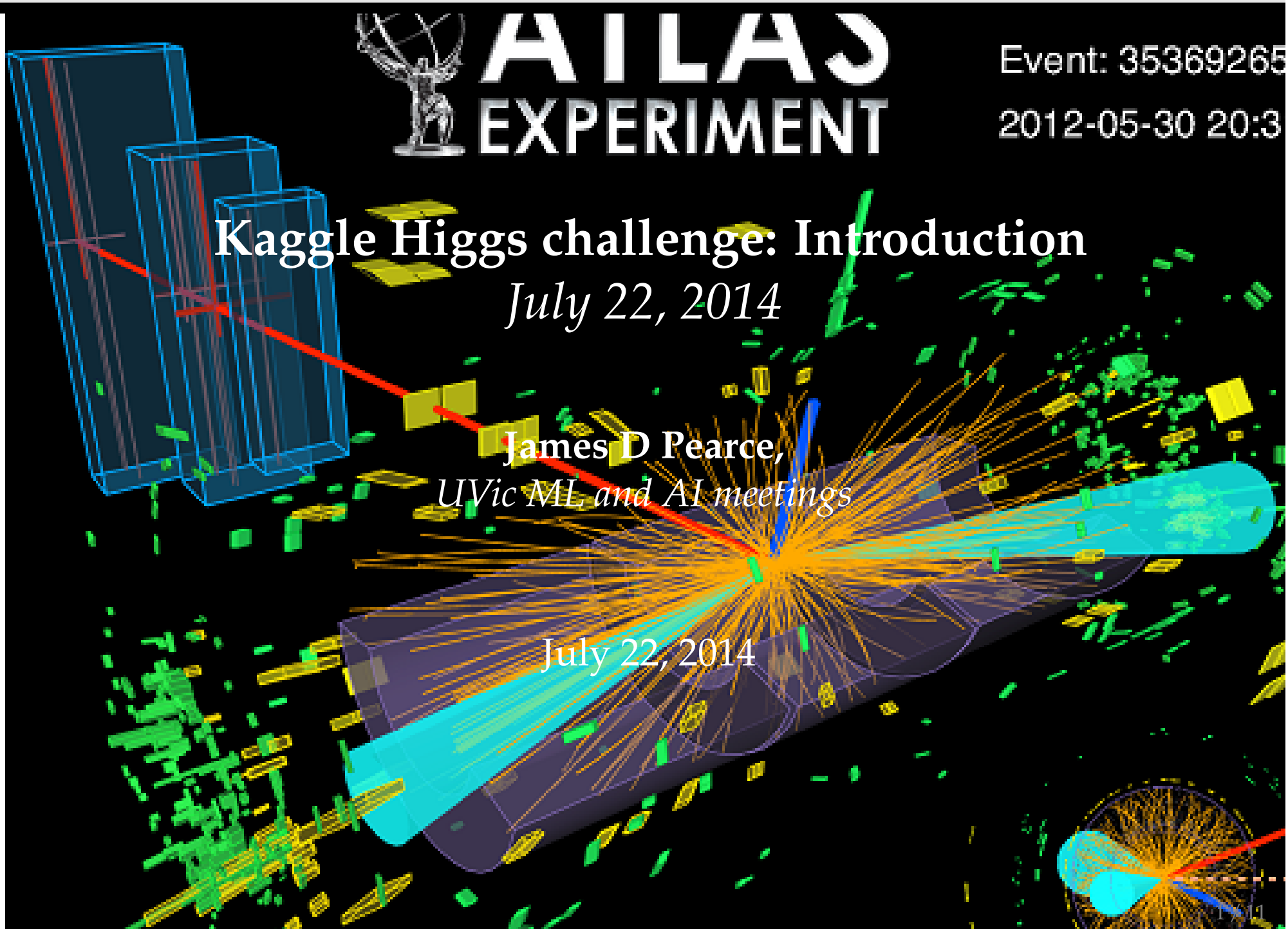


Kaggle Higgs challenge: Introduction

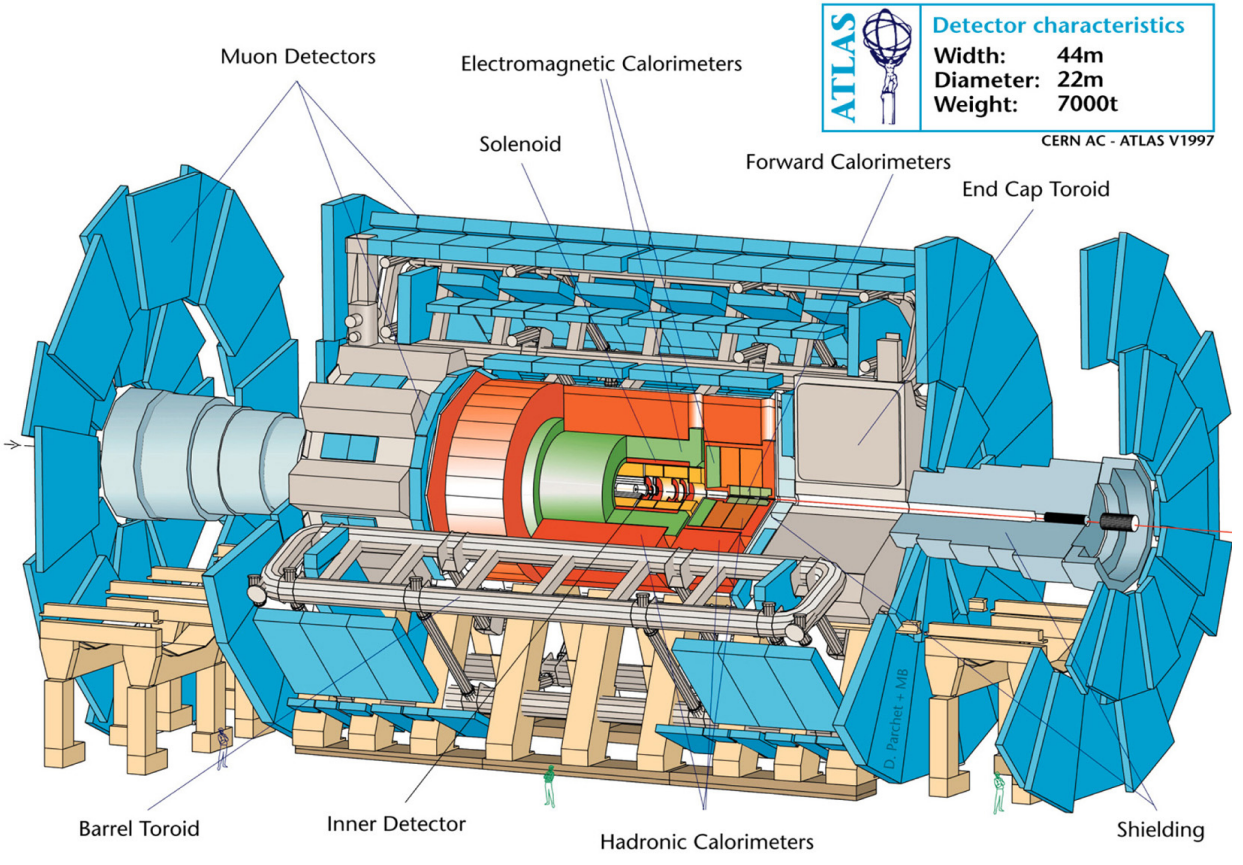
July 22, 2014

James D Pearce,
UVic ML and AI meetings

July 22, 2014

THE ATLAS DETECTOR

The data in the kaggle competition is "real" simulated data, which is heavily used by the ATLAS collaborations and is essentially identical to collected data.



THE DATA

Data consists of a training set of 250,000 events (instances/rows) and a test set of 550,000 events. Kaggle leader board is evaluated on 18% of test set, your final score is evaluated on the full test set when the competition comes to a close.

Training Set

- column 0: event ids $\rightarrow$ only useful for making submission file.
- columns 1-30: features (X)
  - 250,000 $\times$ 30 matrix of floats
  - -999.0 means "object not in event"
- column 31: weights (w)
  - 250,000 element vector of floats
  - give importance of event to AMS metric
- column 32: class label (y)
  - 250,000 element vector of ints (1 or 0)
  - "s" for signal (higgs event), "b" for background (no higgs event).

EVALUATION METRIC

$$AMS = \sqrt{2\left((s+b+br)\log\left(1+\frac{s}{b+br}\right)-s\right)} \tag{1}$$

$$AMS \approx \frac{s}{\sqrt{b}} \text{ if } b \gg s \tag{2}$$

where $b_r = 10$, $s = \sum w_i y_i \hat{y}_i$ and $b = \sum w_i (1 - y_i)\hat{y}_i$. In other words s is the (weighted) number of *true positives* and b is the number of *false positives*. This metric is meant to approximate the significance of the higgs signal against the null hypothesis, i.e. no higgs model.

- $3 \leq AMS < 4$: *statistically significant*
- $4 \leq AMS < 5$: *...very interesting*
- $AMS \geq 5$: *Party time!*

HIGGS_MODEL.PY: LOADING DATA

```
training_dataset = "training.csv"
data = np.array(
    list(csv.reader(open(training_dataset,'rb'), delimiter
    = ','))  )
    X = np.array(data[1:,1:-2], float)

# Make class label and weight vectors
    y_train = np.array([int(row[-1] == 's') for row in
        data[1:]])
    w_train = np.array([float(row[-2]) for row in data[1:]])
```

The python package pandas can make this less painful.

HIGGS_MODEL.PY: SCALING DATA

```
if scale_data:
    imputer =
        preprocessing.Imputer(missing_values=-999.0,
        strategy='mean', axis=0, verbose=0, copy=False)
    imputer.fit_transform(X)
    X_train = preprocessing.scale(np.array(X))
else:
    X_train = X
```

Scaling/normalizing data is only necessary for certain classifier. For others it may actually reduce performance.

HIGGS_MODEL.PY: SPLITTING DATA

```
if do_test:
        X_train, X_test, y_train, y_test, w_train, w_test =
            train_test_split(X_train, y_train, w_train,
            test_size=split_level, random_state=42)
        # weights need to be rescaled for AMS calculation
        w_test *= 1./split_level
        w_train *= 1./(1. - split_level)
```

In general k-fold cross-validation is a much better method for test models

HIGGS_MODEL.PY: TRAINING A CLASSIFIER

```
clf = RandomForestClassifier(n_estimators=10,
    max_depth=None, min_samples_split=1, verbose=2,
    n_jobs=-1)

clf.fit(X_train,y_train)
```

It's very easy to swap in different classifiers with scikit-learn

HIGGS_MODEL.PY: TESTING THE CLASSIFIER

```
if do_test:
    prob = clf.predict_proba(X_test).T[1]
    y_pred = prob > signal_threshold

    print 'Accuracy: ', sum( 1.0 for i in
        range(len(y_test)) if y_test[i] ==
        int(y_pred[i]) )/float(len(y_test))

    ams = AMS(y_test,y_pred,w_test)
    print 'AMS: ', ams
```

POSSIBLE IMPROVEMENTS:

Easy improvements

- Grid search for optimal signal threshold
- Better classifier?
- Optimize hyper-parameters with CV

Intermediate improvements

- Dimension reduction:
  - PCA? $\rightarrow$ probably not
- Dimension augmentation
  - k-means clustering
  - Gaussian mixture models (EM algorithm)
  - kth nearest neighbour

POSSIBLE IMPROVEMENTS:

Advanced improvements

- Use weight information in training algorithm
  - Weight error function with event weights
  - Maximize AMS directly?
  - Find better proxy?
- Feature engineering
  - Razor variables
  - Additional “event shape” variables
- Generate additional data from training set?
- Use unsupervised learning in test + training set for pre-training?
- Use classifier with deep architecture?