Unsupervised Learning and Dimensionality Reduction

March 22, 2015

1 Unsupervised Learning and Dimensionality Reduction

```
In [1]: %matplotlib inline
```

```
In [132]: %%javascript
          IPython.load_extensions('calico-spell-check')
```

```
<IPython.core.display.Javascript object>
```

The goal is for you to think about how these algorithms are the same as, different from, and interact with your earlier work.

1.0.1 Classification Data Review

```
In [33]: from algo_evaluation import datasets
```

Higgs Data quick review:

- given outcomes of particle decays, detect Higgs boson;
- most of the **supervised** algorithms gave acceptable accuracy ranging from **0.8 - 0.9** far outperforming the **backpropagation** algorithm for neural networks (**0.5**)
- **learning weights** with Randomized Optimization algorithms improved accuracy of the neural network to **0.7** and higher

Additional Changes:

- features were rescaled as a requirement to feature transformation alogrithms
- previously manually pruned features were put back to allow feature selection algorithms to autotically choose the most informative ones
- dataset was not reduced in size since dimensionality reduction provides speed advantage (the main reason for sampling records in the previous analysis)

```
In [14]: higgs_data = datasets.load_higgs_train(sample_size=None, verbose=True, scale=True,
                                                 prune_features=False)
         all_higgs_features = higgs_data[0].shape[1]
```

```
Size of the dataset: 68114
Number of features: 30
Number of positives (signal): 31894
Number of negatives (background): 36220
```

Converters Data quick review:

- given online user behavior and preferences predict whether or not ad display will result in convertion;
- in contrast to Higgs dataset, supervized algorithm all provided high accuracy >94%
- this dataset was not subjected to randomized optimization analysis

```
In [36]: bid_data = datasets.load_bidding_train(verbose=True, scale=True)
```

```
Size of the dataset: 57970
Number of features: 15
Number of converters: 399
Number of non-converters: 54615
Number of leads: 2956
```

```
/Users/maestro/anaconda/lib/python2.7/site-packages/sklearn/preprocessing/data.py:145: DeprecationWarni
  Xr -= mean_
/Users/maestro/anaconda/lib/python2.7/site-packages/sklearn/preprocessing/data.py:147: DeprecationWarni
  Xr /= std_
```

1.0.2 K-means Clustering

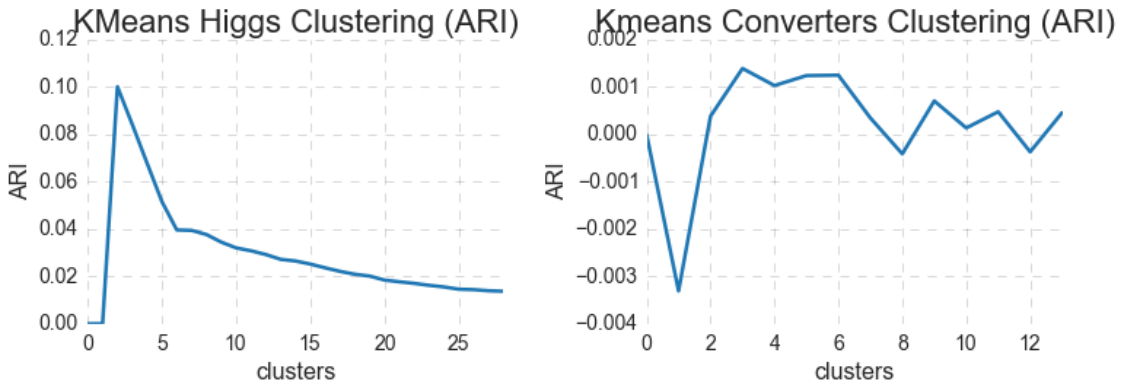
Evaluating cluster performance is different from the techiques used in Supervised Algorithms where accuracy was estimated roughly as a count of wrong answers. For clustering we must imploy similarity metric: did clustering define separations of data similar to ground truth. In this analysis I shall use **Adjusted Rand Index** that measures the similarity of the two assignments, ignoring permutations and with chance normalization.

```
In [99]: from algo_evaluation.clustering import kmeans_eval
         from algo_evaluation.clustering import plot_cluster_estimation
```

```
In [9]: df_kmeans_higgs = kmeans_eval.estimate_clusters(higgs_data)
```

```
In [37]: df_kmeans_converters = kmeans_eval.estimate_clusters(bid_data)
```

```
In [95]: plot_cluster_estimation.plot_kmeans_cluster_score(df_kmeans_higgs, df_kmeans_converters)
```



Observations:

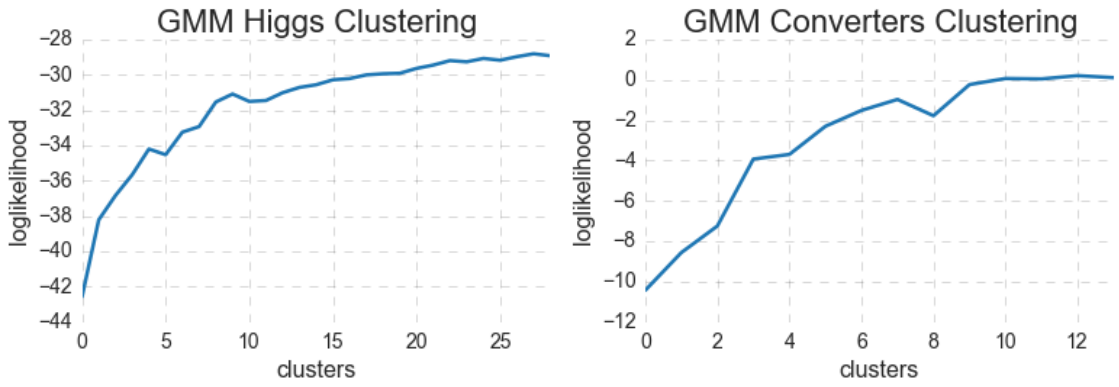
Best data separation for **Higgs** data achieved with number of clusters equal **2** which makes sense since there are two classes in the data: signal and background. As number of the clusters grow, similarity gradually approaches zero.

Similarly, number of clusters equal **3** for **Converters** data achieves best separation for the three classes: converters, non-converters and leads.

Note, however, that neither of the above data separation is considered good (ARI=1.0 is a perfect score). Since scores are close to 0.0, label assignment is rather uniform especially for converters data. This suggests that datasets have more complex structure and are not easily separable into groups.

1.0.3 Expectation Maximization

```
In [96]: from algo_evaluation.clustering import gmm_eval

In [80]: df_gmm_higgs = gmm_eval.estimate_clusters(higgs_data)

In [82]: df_gmm_converters = gmm_eval.estimate_clusters(bid_data)

In [100]: plot_cluster_estimation.plot_gmm_cluster_score(df_gmm_higgs, df_gmm_converters)
```



1.0.4 Feature Selection

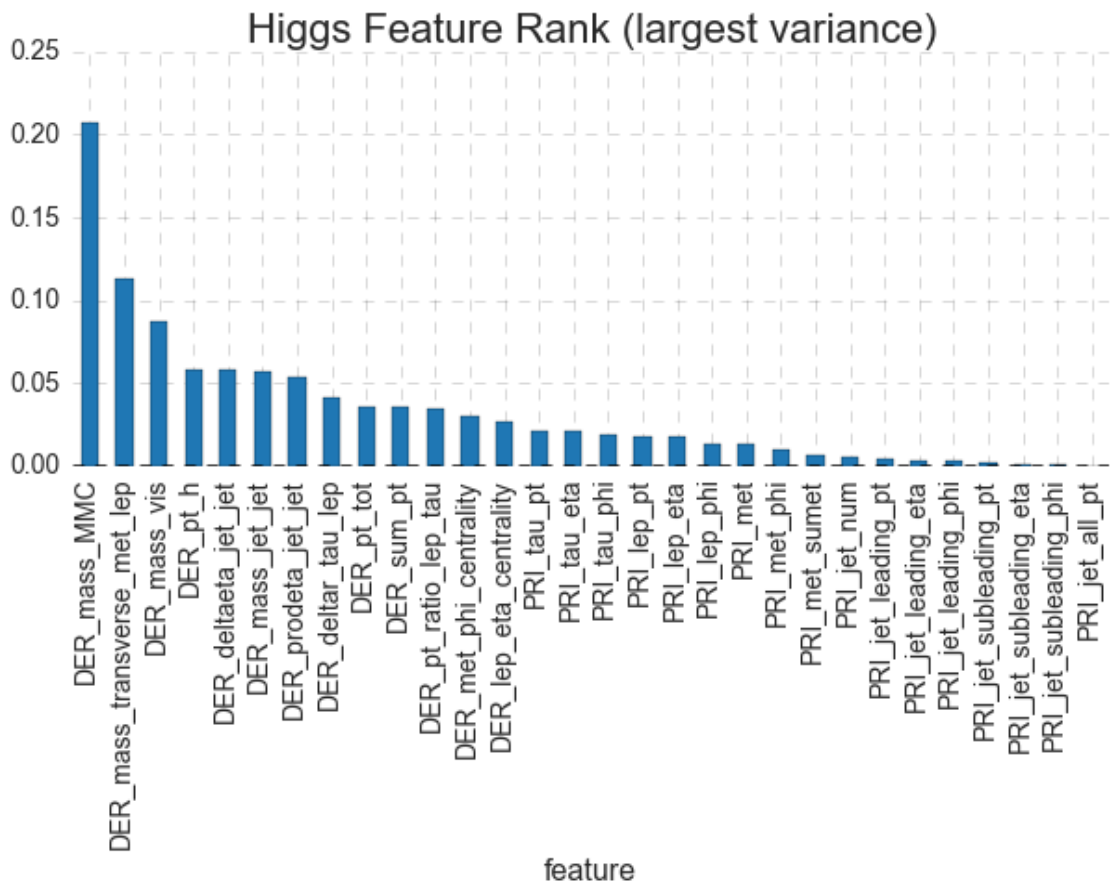
In the previous analysis Higgs records were sampled and features were manually pruned due to the slow speed of supervized algorithms. In this analysis, complete dataset is restored to take advantage of automatic feature selection using dimensionality reduction algorithms.

Dimensionality reduction is the task of deriving a set of features that is smaller than the original feature set while retaining most of the variance of the original data. To estimate the best number of components to be used for data transformation, I used PCA and ExtraTreeClassifier to rank the features and choose what seems to be optimal.

```
In [106]: from algo_evaluation.feature_selection import pca_eval, extra_tree_eval, rand_projections

In [81]: pca_rank = pca_eval.rank_features(higgs_data, n_components=all_higgs_features)

In [ ]: extra_tree_rank = extra_tree_eval.rank_features(higgs_data, n_estimators=all_higgs_features)
```

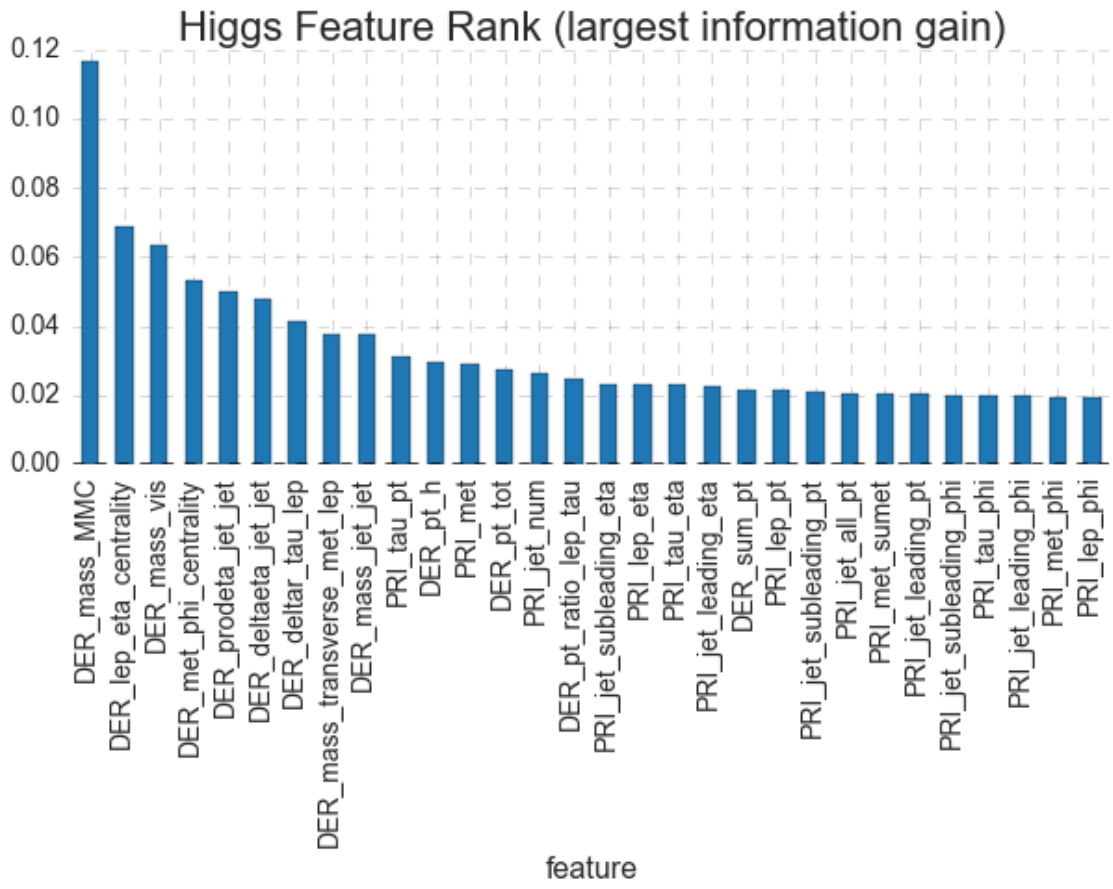
Using **PCA**, I extracted variances for each feature and sorted them from largest to smallest. Visualizing the results made it clear that "Derived" features have larger spread than "Primitives" and thus are more informative for classification. This supports the decision made in the previous analysis where "Primitive" features were pruned based on feature by feature analysis.

```
In [82]: pca_eval.plot_rank(pca_rank)
```



Visual inspection of the PCA feature ranking suggests that the best number of components could be selected as **7**. To confirm this selection I applied **ExtraTreesClassifier**, which implements a meta estimator that fits a number of randomized decision trees on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. Note that both algorithms ranked DER_mass_MMC the highest which makes sense since this feature estimates mass of the Higgs boson candidate and indeed is the most informative.

```
In [24]: extra_tree_eval.plot_rank(extra_tree_rank)
```



1.0.5 Component Number Estimation

```
In [123]: reload(pca_eval)
          pca_comp = pca_eval.estimate_components(higgs_data)

In [107]: rp_comp = rand_projections.estimate_components(higgs_data)
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-107-a7730463bcc0> in <module>()
----> 1 rp_comp = rand_projections.estimate_components(higgs_data)


/Users/maestro/schoolspace/bag-of-algorithms/algo_evaluation/feature_selection/rand_projections.
     44     for n in range(n_components):
     45         estimator.n_components = n
---> 46         score = np.mean(cross_val_score(estimator, features))
     47         pca_scores.append([n, score])
     48     df = pd.DataFrame.from_records(pca_scores, columns=['components', 'score'])
```

```
/Users/maestro/anaconda/lib/python2.7/site-packages/sklearn/cross_validation.pyc in cross_val_sc
   1141
   1142     cv = _check_cv(cv, X, y, classifier=is_classifier(estimator))
-> 1143     scorer = check_scoring(estimator, score_func=score_func, scoring=scoring)
   1144     # We clone the estimator to make sure that all the folds are
   1145     # independent, and that it is pickle-able.


/Users/maestro/anaconda/lib/python2.7/site-packages/sklearn/metrics/scorer.pyc in check_scoring(
    236         raise TypeError(
    237             "If no scoring is specified, the estimator passed should "
--> 238             "have a 'score' method. The estimator %r does not." % estimator)
    239     else:
    240         raise TypeError(


TypeError: If no scoring is specified, the estimator passed should have a 'score' method. The es
        n_components=0, random_state=None) does not.
```

In [91]:
```
ica_comp = ica_eval.estimate_components(higgs_data)
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-91-ffb219b27a78> in <module>()
----> 1 ica_comp = ica_eval.estimate_components(higgs_data)


/Users/maestro/schoolspace/bag-of-algorithms/algo_evaluation/feature_selection/ica_eval.py in es
     26     for n in range(n_components):
     27         estimator.n_components = n
---> 28         score = np.mean(cross_val_score(estimator, features))
     29         pca_scores.append([n, score])
     30     df = pd.DataFrame.from_records(pca_scores, columns=['components', 'score'])


/Users/maestro/anaconda/lib/python2.7/site-packages/sklearn/cross_validation.pyc in cross_val_sc
   1141
   1142     cv = _check_cv(cv, X, y, classifier=is_classifier(estimator))
-> 1143     scorer = check_scoring(estimator, score_func=score_func, scoring=scoring)
   1144     # We clone the estimator to make sure that all the folds are
   1145     # independent, and that it is pickle-able.


/Users/maestro/anaconda/lib/python2.7/site-packages/sklearn/metrics/scorer.pyc in check_scoring(
    236         raise TypeError(
    237             "If no scoring is specified, the estimator passed should "
--> 238             "have a 'score' method. The estimator %r does not." % estimator)
    239     else:
    240         raise TypeError(
```

```
TypeError: If no scoring is specified, the estimator passed should have a 'score' method. The es
n_components=0, random_state=None, tol=0.0001, w_init=None,
whiten=True) does not.
```

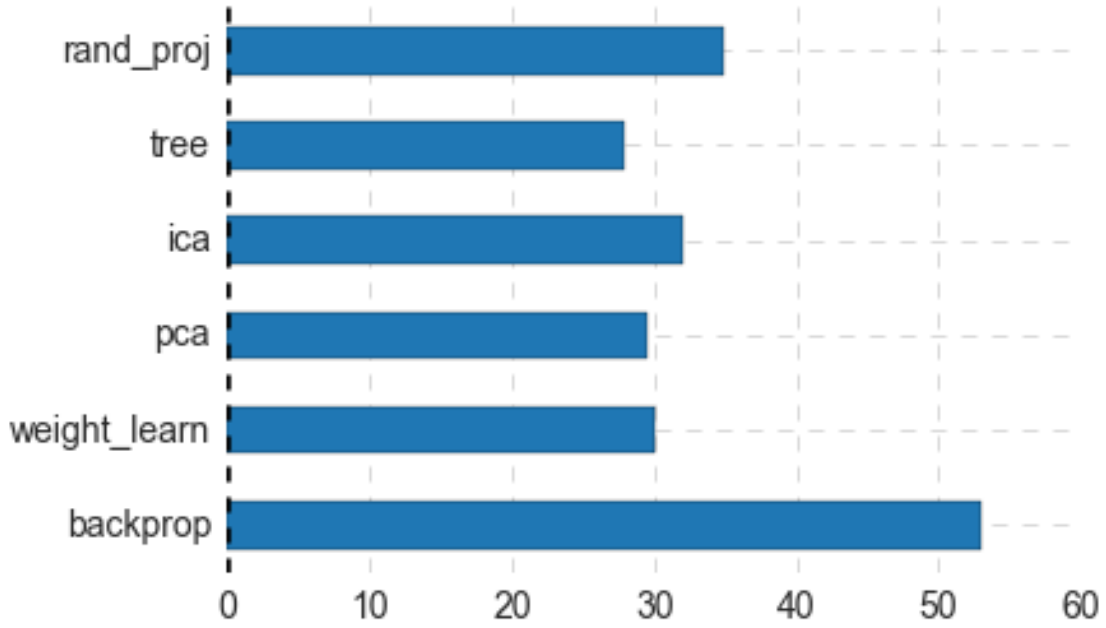
1.0.6 Feature Transformation

Having estimated number of componets, now I can actually run the transformation to reduce the dimension of the Higgs data.

```
In [147]: from algo_evaluation.feature_selection import rand_projections, ica_eval

In [113]: pca_trns, pca_elapsed = pca_eval.transform(higgs_data, n_components=7)

In [114]: rand_proj_trns, rand_proj_elapsed = rand_projections.transform(higgs_data, n_components=7)

In [117]: ica_trns, ica_elapsed = ica_eval.transform(higgs_data, n_components=7)

In [118]: tree_trns, tree_elapsed = extra_tree_eval.transform(higgs_data, n_components=7)
```

1.0.7 Neural Networks post Dimensionality Reduction

```
In [110]: from algo_evaluation.supervised import neural_network as nn

In [142]: higgs_backprop_err = 100 * (1 - 0.47)
          print 'Backpropagation training error =', higgs_backprop_err
          higgs_weight_learn_err = 100 * (1 - 0.7)
          print 'Weights learning training error =', higgs_weight_learn_err

Backpropagation training error = 53.0
Weights learning training error = 30.0

In [112]: def reduced_higgs(transformed):
              return {'features': transformed, 'weights':higgs_data[1], 'labels': higgs_data[2]}

In [119]: _, _, nn_higs_pca_error = nn.run_neural_net(reduced_higgs(pca_trns))

In [120]: _, _, nn_higs_ica_error = nn.run_neural_net(reduced_higgs(ica_trns))

In [121]: _, _, nn_higs_tree_error = nn.run_neural_net(reduced_higgs(tree_trns))

In [122]: _, _, nn_higs_rp_error = nn.run_neural_net(reduced_higgs(rand_proj_trns))

In [150]: nn_higgs_data = pd.Series([backprop_err,
                                     weight_learn_err,
                                     nn_higs_pca_error,
                                     nn_higs_ica_error,
                                     nn_higs_tree_error,
                                     nn_higs_rp_error],
                                    index=['backprop','weight_learn', 'pca', 'ica', 'tree', 'rand_proj']
                                   name='trnerr')
          nn_higgs_data.plot(kind='barh', figsize=(6,4))

Out[150]: <matplotlib.axes._subplots.AxesSubplot at 0x10fe4cb10>
```



1.0.8 Clusteing post Dimensionality Reduction

In [146]:
```
df_kmeans_higgs_dim = kmeans_eval.estimate_clusters(reduced_higgs(pca_trns))
```

In [145]:
```
df_kmeans_converters_dim = kmeans_eval.estimate_clusters(reduced_converted(pca_trns))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)

<ipython-input-145-a95465733a50> in <module>()
----> 1 df_kmeans_converters_dim = kmeans_eval.estimate_clusters(reduced_converted(pca_trns))


/Users/maestro/schoolspace/bag-of-algorithms/algo_evaluation/clustering/kmeans_eval.py in estima
     72     for n in range(1, n_clusters):
     73         estimator.n_clusters = n
---> 74         score = np.mean(cross_val_score(estimator, features, labels, scoring='adjusted_rand_s
     75         scores.append([n, score])
     76     df = pd.DataFrame.from_records(scores, columns=['clusters', 'score'])


/Users/maestro/anaconda/lib/python2.7/site-packages/sklearn/cross_validation.pyc in cross_val_sc
   1138     """
   1139     X, y = check_arrays(X, y, sparse_format='csr', allow_lists=True,
-> 1140                         allow_nans=True, allow_nd=True)
   1141
   1142     cv = _check_cv(cv, X, y, classifier=is_classifier(estimator))
```

```
/Users/maestro/anaconda/lib/python2.7/site-packages/sklearn/utils/validation.pyc in check_arrays
252         if size != n_samples:
253             raise ValueError("Found array with dim %d. Expected %d"
--> 254                              % (size, n_samples))
255
256         if not allow_lists or hasattr(array, "shape"):


ValueError: Found array with dim 57970. Expected 68114
```

You are to run a number of experiments. Come up with at least two datasets
Run the clustering algorithms on the data sets and describe what you see
Apply the dimensionality reduction algorithms to the two datasets and describe what you see.
Reproduce your clustering experiments, but on the data after you've run dimensionality reduction on it.
Apply the dimensionality reduction algorithms to one of your datasets from assignment #1
and rerun your neural network learner on the newly projected data

Apply the clustering algorithms to the same dataset to which you just applied the dimensionality reduction algorithms (you've probably already done this), treating the clusters as if they were new features. In other words, treat the clustering algorithms as if they were dimensionality reduction algorithms. Again, rerun your neural network learner on the newly projected data.

A discussion of your datasets, and why they're interesting: If you're using the same datasets as before at least briefly remind us of what they are so we don't have to revisit your old assignment write-up. explanations of your methods: How did you choose k? a description of the kind of clusters that you got. analyses of your results. Why did you get the clusters you did? Do they make "sense"? If you used data that already had labels (for example data from a classification problem from assignment #1) did the clusters line up with the labels? Do they otherwise line up naturally? Why or why not? Compare and contrast the different algorithms. What sort of changes might you make to each of those algorithms to improve performance? How much performance was due to the problems you chose? Be creative and think of as many questions you can, and as many answers as you can. Take care to justify your analysis with data explictly. Can you describe how the data look in the new spaces you created with the various aglorithms? For PCA, what is the distribution of eigenvalues? For ICA, how kurtotic are the distributions? Do the projection axes for ICA seem to capture anything "meaningful"? Assuming you only generate k projections (i.e., you do dimensionality reduction), how well is the data reconstructed by the randomized projections? PCA? How much variation did you get when you re-ran your RP several times (I know I don't have to mention that you might want to run RP many times to see what happens, but I hope you forgive me)? When you reproduced your clustering experiments on the datasets projected onto the new spaces created by ICA, PCA and RP, did you get the same clusters as before? Different clusters? Why? Why not? When you re-ran your neural network algorithms were there any differences in performance? Speed? Anything at all? It might be difficult to generate the same kinds of graphs for this part of the assignment as you did before; however, you should come up with some way to describe the kinds of clusters you get. If you can do that visually all the better.

In [ ]: