SciComp with Py

Support Vector Machines

Vladimir Kulyukin
Department of Computer Science
Utah State University

Kernel Tricks

- When operating in a high-dimensional, implicit feature space (classes may be known but features are unknown), it is expensive to compute the explicit coordinates of each feature vector
- It is computationally cheaper to define a metric (e.g., inner product) for computing similarity between each pair of feature vectors
- This approach is called the "kernel trick"; such kernel functions have been developed for graphs, texts, and images
- One can play kernel tricks on any data that can be represented as feature vectors

Hyperplanes

- Ambient space is a space that surrounds an object
- A hyperplane is a subspace of one dimension less than its ambient space
- Examples:

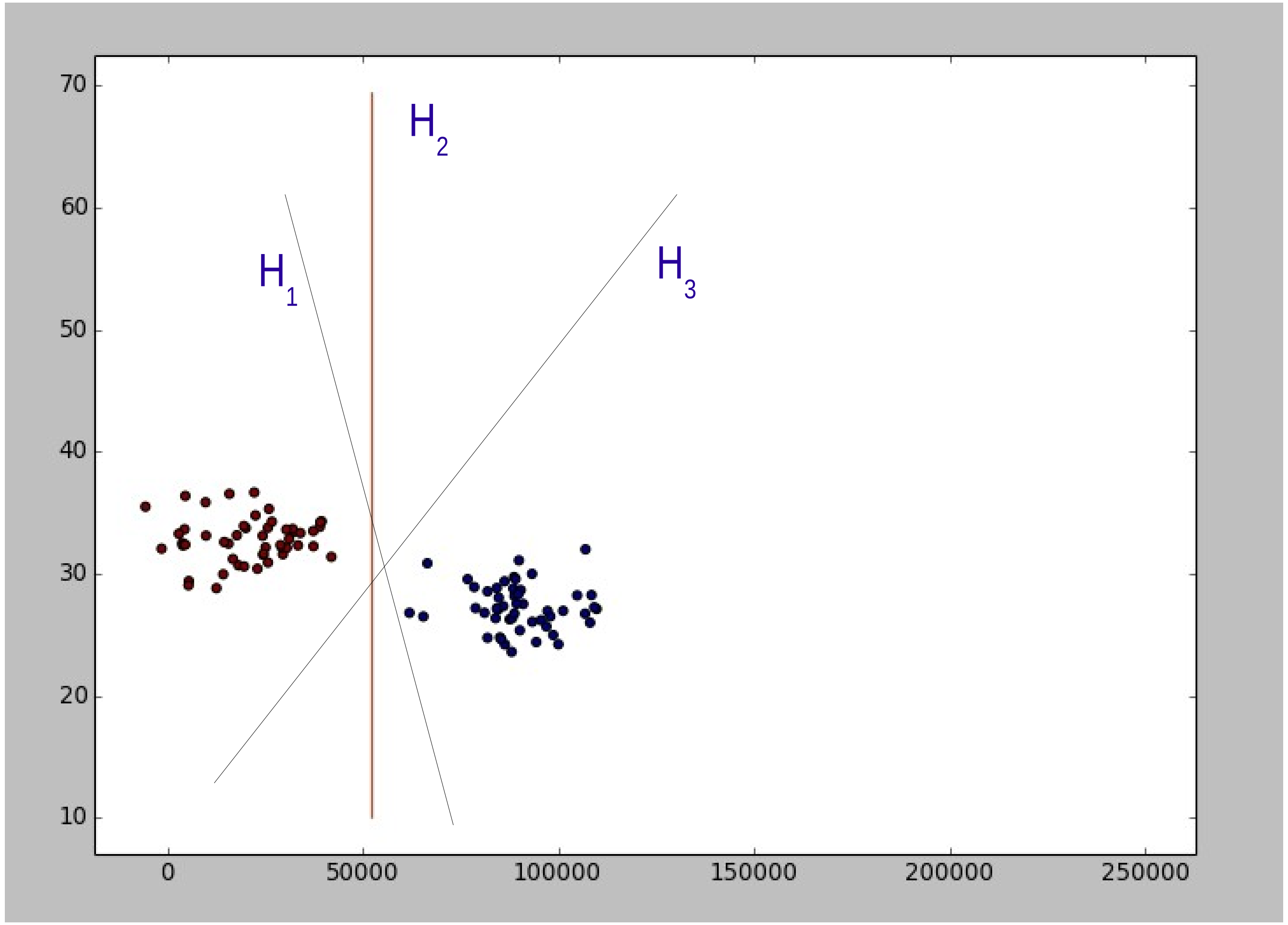
  - In 2D spaces, hyperplanes are 1D lines

  - In 3D spaces, hyperplanes are 2D planes

Hyperplane Separation

- All data points are vectors in n-dimensional spaces
- Each data point is of a known class (this means that SVM is a supervised learning algorithm)
- The objective of the SVM learning algorithm is to find n-1 dimensional hyperplanes that best separate the data points into clusters
- There may be many hyperplanes that separate the data
- The best hyperplanes separate the data by the widest margin

Hyperplane Separation Example in 2D



All three hyperplanes (lines) separate the two clusters but H_1 and H_3 do not separate them by as wide a margin as H_2

Largest Margin Criterion

- To classify data points into clusters, an SVM constructs a set of hyperplanes in a high-dimensional space
- The best separation is achieved by a hyperplane with the largest distance to the nearest training data point of any class (so-called functional margin)
- In general, the larger the margin the lower the generalization error of the classification

SVM Application Domains

- Image classification, especially image segmentation where image pixels are separated into clusters
- Optical and handwritten character recognition
- Classification of multi-dimensional datasets in social and biological sciences
- Text classification

Common SVM Kernels

- Linear Kernels – use lines to compute similarity among and/or to separate data points
- Polynomial Kernels – use polynomials to compute similarity among and/or to separate data points
- RBF (Radial Basis Function) Kernels – use circular functions to compute similarity among and/or to separate data points

Problem

Let's train a decision tree for the IRIS dataset by using only the first 2 features of each data item. Let's also print the classifier classification reports and confusion matrices for all four classifiers and compare them.

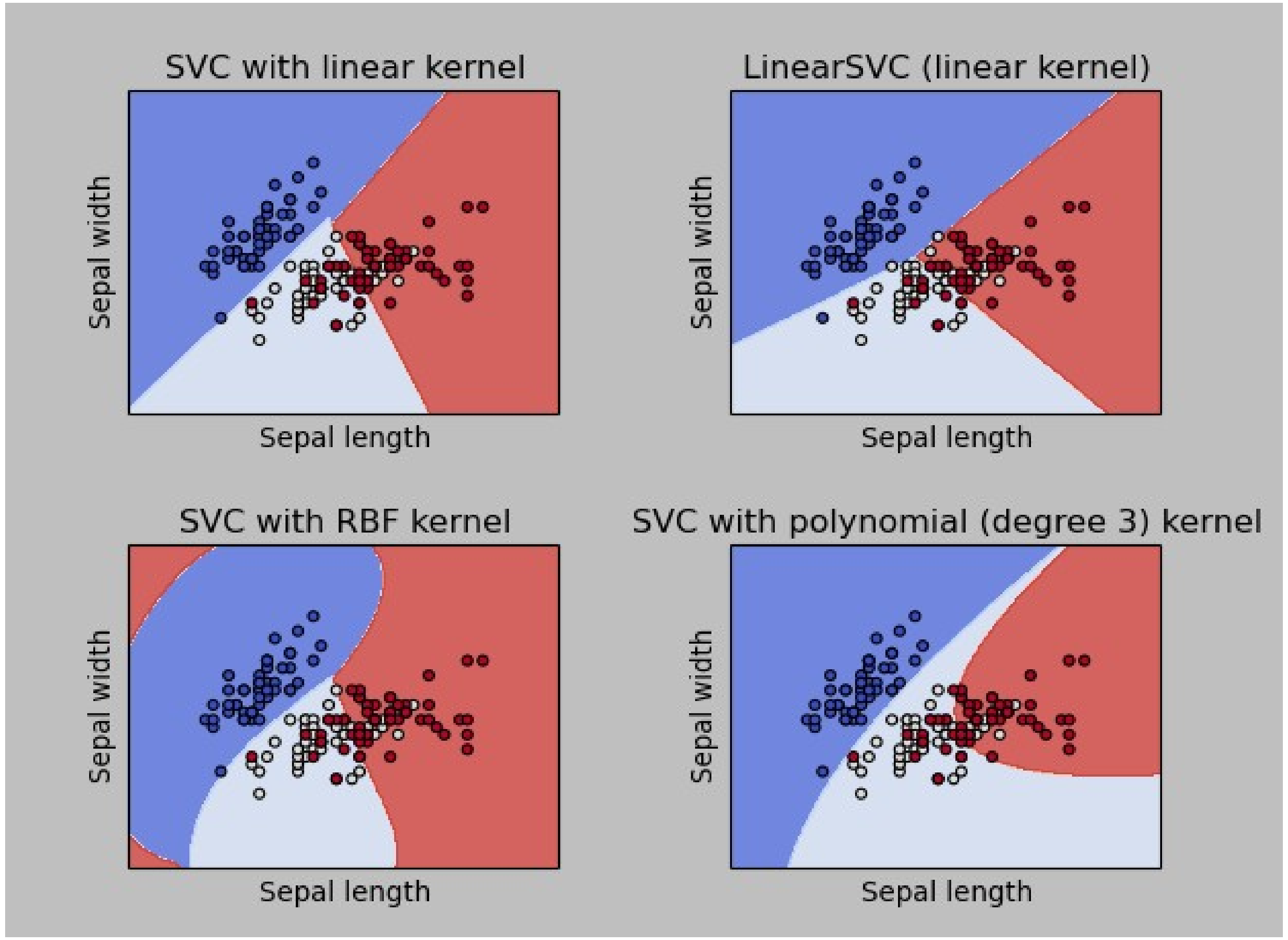
Solution: Plotting Generated Data & Training Linear SVM

```
iris = datasets.load_iris()
# we only take the first two features: sepal length and sepal width
X = iris.data[:, :2]
y = iris.target

# create four types of svm classifiers
C = 1.0  # SVM regularization parameter
svc = svm.SVC(kernel='linear', C=C).fit(X, y)
rbf_svc = svm.SVC(kernel='rbf', gamma=0.7, C=C).fit(X, y)
poly_svc = svm.SVC(kernel='poly', degree=3, C=C).fit(X, y)
lin_svc = svm.LinearSVC(C=C).fit(X, y)
```

source in svm_2d_iris.py

2D Iris Dataset SVM Separation with 4 Kernels



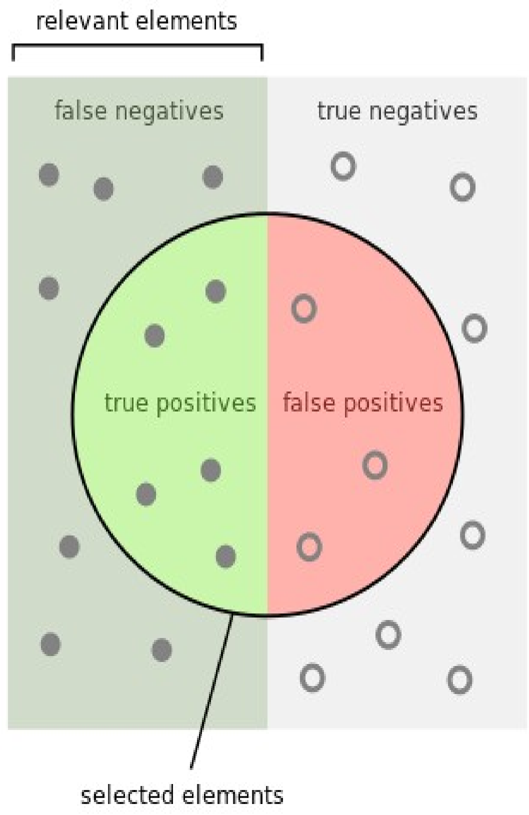
source code in svm_2d_iris.py

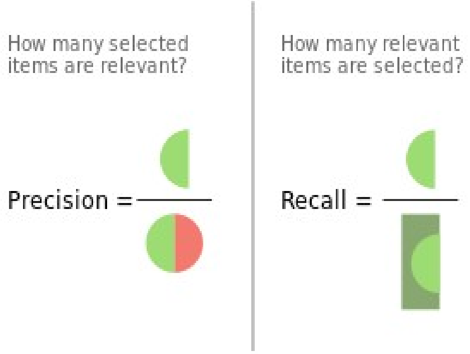
Problem

Train a decision tree for the IRIS dataset. Train 3 SVM classifiers (linear, 3rd deg poly, and RBF) for IRIS. Print the classifier classification reports and confusion matrices for all four classifiers and compare them.

Review: Recall vs Precision

- Precision is percentage of retrieved data points that are relevant
- Recall is percentage of relevant data points that are retrieved
- Image source: https://en.wikipedia.org/wiki/Precision_and_recall





Review: F1 Support

$$F_1 = 2 \cdot \frac{1}{\frac{1}{\mathrm{recall}} + \frac{1}{\mathrm{precision}}} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

F1 metric combines both precision and recall

Solution: Decision Tree Classification Report for IRIS

	precision	recall	f1-score	support
0	1.00	1.00	1.00	9
1	1.00	0.77	0.87	22
2	0.74	1.00	0.85	14
avg / total	0.92	0.89	0.89	45

number of occurrences of each cluster label in the target vector (ground truth)

Confusion matrix:

```
[[ 9  0  0]
 [ 0 17  5]
 [ 0  0 14]]
```

Look at row 1. There are 17 label 1 (versicolor) items accurately classified as label 1;
The are 5 label 1 items inaccurately classified as label 2 (virginica).
So, recall for label 1 is 17/(17+5) = 0.77.
Look at column 1. Of the items classified as label 1, no other items were classified as label 1. So, precision for label 1 is 17/17 = 1.0.
Look at column 2. Of the items classified as label 2 (virginica), 5 items were classified as Label 1 (versicolor). So, precision is 14/(14+5) = 0.74.

source in iris_decision_tree.py

Solution: Creating 3 SVM Classifiers for IRIS

```
## get the data, the data items, and target
iris_data = datasets.load_iris()
data_items = iris_data.data
data_target = iris_data.target

## let's create 3 classifiers
C = 1.0  # SVM regularization error parameter
# an svm classifier with linear kernel
lin_svc = svm.SVC(kernel='linear', C=C)
# an svm classifier with rbf kernel
rbf_svc = svm.SVC(kernel='rbf', C=C)
# an svm classifier with 3rd degree poly kernel
poly_svc = svm.SVC(kernel='poly', degree=3, C=C)
```

source in svm_4d_iris.py

Solution: Training Classifiers & Generating Reports

```
def print_svc_report(clf, data_items, data_target, test_size=0.3):
    #1. get train & test data and train and test targets
    train_data, test_data, train_target, test_target = \
                train_test_split(data_items,
                                 data_target,
                                 test_size=test_size,
                                 random_state=random.randint(0, 1000))
    # 2. fit the classifier over the train data and train target
    clf.fit(train_data, train_target)
    # 3. set the ground truth for test target vector
    clf_expected = test_target
    # 4. test the classifier on test data
    clf_predicted = clf.predict(test_data)
    # 5. print the classification report followed by confustion matrix
    print("Classification report for SVC with linear kernel %s:\n%s\n"
        % (clf, metrics.classification_report(clf_expected, clf_predicted)))
    print("Confusion matrix:\n%s" % metrics.confusion_matrix(clf_expected, clf_predicted))
    print('------------------------------------------------')
```

source in svm_4d_iris.py

Solution: 3 SVMs on IRIS Dataset Side by Side

Linear SVM

	precision	recall	f1-score	support
0	1.00	1.00	1.00	14
1	0.86	1.00	0.92	12
2	1.00	0.89	0.94	19
avg / total	0.96	0.96	0.96	45

Confusion matrix:

```
[[14  0  0]
 [ 0 12  0]
 [ 0  2 17]]
```

RBF SVM

	precision	recall	f1-score	support
0	1.00	1.00	1.00	17
1	1.00	0.94	0.97	16
2	0.92	1.00	0.96	12
avg / total	0.98	0.98	0.98	45

Confusion matrix:

```
[[17  0  0]
 [ 0 15  1]
 [ 0  0 12]]
```

3rd Deg Poly SVM

	precision	recall	f1-score	support
0	1.00	1.00	1.00	16
1	1.00	0.90	0.95	10
2	0.95	1.00	0.97	19
avg / total	0.98	0.98	0.98	45

Confusion matrix:

```
[[16  0  0]
 [ 0  9  1]
 [ 0  0 19]]
```

All SVM classifiers are in the same ballpark

Solution: Decision Tree vs. SVM on IRIS Dataset

Decision Tree

	precision	recall	f1-score	support
0	1.00	1.00	1.00	9
1	1.00	0.77	0.87	22
2	0.74	1.00	0.85	14
avg / total	0.92	0.89	0.89	45

Confusion matrix:

```
[[ 9  0  0]
 [ 0 17  5]
 [ 0  0 14]]
```

3rd Deg Poly SVM

	precision	recall	f1-score	support
0	1.00	1.00	1.00	16
1	1.00	0.90	0.95	10
2	0.95	1.00	0.97	19
avg / total	0.98	0.98	0.98	45

Confusion matrix:

```
[[16  0  0]
 [ 0  9  1]
 [ 0  0 19]]
```

Which classifier is better? Looks like SVM with the 3rd deg poly kernel is much better than the decision tree.

Problem

Train a decision tree classifier for the DIGITS dataset and print its classification report. Train 3 SVM classifiers (linear, 3rd deg poly, and RBF) for the DIGITS dataset. Print the classifier classification reports and confusion matrices for all three. Which classifier is best for this dataset?

Solution: Digits Decision Tree Classification Report

```
             precision    recall  f1-score   support

          0       1.00      0.90      0.95        50
          1       0.82      0.77      0.80        53
          2       0.85      0.89      0.87        56
          3       0.86      0.80      0.83        60
          4       0.84      0.84      0.84        62
          5       0.90      0.81      0.85        53
          6       0.91      0.91      0.91        47
          7       0.92      0.89      0.90        61
          8       0.73      0.78      0.75        49
          9       0.66      0.84      0.74        49

avg / total       0.85      0.84      0.84       540

Confusion matrix:
[[45  0  2  0  0  2  0  0  1  0]
 [ 0 41  3  1  1  0  1  2  4  0]
 [ 0  0 50  2  1  0  1  1  1  0]
 [ 0  1  0 48  0  0  0  1  1  9]
 [ 0  1  0  0 52  1  1  1  3  3]
 [ 0  1  1  1  1 43  1  0  2  3]
 [ 0  0  1  0  2  1 43  0  0  0]
 [ 0  0  0  1  3  0  0 54  1  2]
 [ 0  3  2  1  1  0  0  0 38  4]
 [ 0  3  0  2  1  1  0  0  1 41]]
```

source in digits_decision_tree.py

Solution: 3 SVM Classifiers for DIGITS Dataset

```
## get the data, the data items, and target
digits_data = datasets.load_digits()
data_items = iris_data.data
data_target = iris_data.target

## let's create 3 classifiers
C = 1.0  # SVM regularization error parameter
# an svm classifier with linear kernel
lin_svc = svm.SVC(kernel='linear', C=C)
# an svm classifier with rbf kernel
rbf_svc = svm.SVC(kernel='rbf', C=C)
# an svm classifier with 3rd degree poly kernel
poly_svc = svm.SVC(kernel='poly', degree=3, C=C)
```

source in svm_digits.py

Solution: Printing Classificaiton Reports

```
if __name__ == '__main__':
    print_svc_report(lin_svc, data_items, data_target)
    print_svc_report(rbf_svc, data_items, data_target)
    print_svc_report(poly_svc, data_items, data_target)
```

source in svm_digits.py

Solution: 3 SVMs on DIGITS Dataset Side by Side

Linear SVM

	precision	recall	f1-score	support
0	1.00	1.00	1.00	49
1	0.90	1.00	0.94	60
2	1.00	1.00	1.00	61
3	1.00	0.98	0.99	48
4	0.98	1.00	0.99	54
5	0.96	1.00	0.98	45
6	1.00	0.98	0.99	63
7	1.00	0.98	0.99	52
8	1.00	0.85	0.92	54
9	0.98	1.00	0.99	54
avg / total	0.98	0.98	0.98	540

Confusion matrix:

```
[[49  0  0  0  0  0  0  0  0  0]
 [ 0 60  0  0  0  0  0  0  0  0]
 [ 0  0 61  0  0  0  0  0  0  0]
 [ 0  0  0 47  0  0  0  0  0  1]
 [ 0  0  0  0 54  0  0  0  0  0]
 [ 0  0  0  0  0 45  0  0  0  0]
 [ 0  1  0  0  0  0 62  0  0  0]
 [ 0  0  0  0  1  0  0 51  0  0]
 [ 0  6  0  0  0  2  0  0 46  0]
 [ 0  0  0  0  0  0  0  0  0 54]]
```

RBF SVM

	precision	recall	f1-score	support
0	1.00	0.25	0.40	56
1	1.00	0.16	0.28	61
2	1.00	0.05	0.09	63
3	1.00	0.15	0.25	48
4	1.00	0.18	0.31	55
5	1.00	0.07	0.14	54
6	1.00	0.37	0.54	57
7	1.00	0.05	0.09	62
8	0.07	1.00	0.13	31
9	1.00	0.08	0.14	53
avg / total	0.95	0.20	0.24	540

Confusion matrix:

```
[[14  0  0  0  0  0  0  0 42  0]
 [ 0 10  0  0  0  0  0  0 51  0]
 [ 0  0  3  0  0  0  0  0 60  0]
 [ 0  0  0  7  0  0  0  0 41  0]
 [ 0  0  0  0 10  0  0  0 45  0]
 [ 0  0  0  0  0  4  0  0 50  0]
 [ 0  0  0  0  0  0 21  0 36  0]
 [ 0  0  0  0  0  0  0  3 59  0]
 [ 0  0  0  0  0  0  0  0 31  0]
 [ 0  0  0  0  0  0  0  0 49  4]]
```

3rd Deg Poly SVM

	precision	recall	f1-score	support
0	1.00	1.00	1.00	52
1	0.99	1.00	0.99	70
2	1.00	1.00	1.00	53
3	0.98	1.00	0.99	49
4	1.00	1.00	1.00	57
5	0.97	0.97	0.97	58
6	1.00	0.98	0.99	57
7	1.00	1.00	1.00	46
8	0.98	0.96	0.97	49
9	0.94	0.94	0.94	49
avg / total	0.99	0.99	0.99	540

Confusion matrix:

```
[[52  0  0  0  0  0  0  0  0  0]
 [ 0 70  0  0  0  0  0  0  0  0]
 [ 0  0 53  0  0  0  0  0  0  0]
 [ 0  0  0 49  0  0  0  0  0  0]
 [ 0  0  0  0 57  0  0  0  0  0]
 [ 0  0  0  0  0 56  0  0  0  2]
 [ 0  0  0  0  0  1 56  0  0  0]
 [ 0  0  0  0  0  0  0 46  0  0]
 [ 0  1  0  0  0  0  0  0 47  1]
 [ 0  0  0  1  0  1  0  0  1 46]]
```

Solution: Decision Tree vs SVM on DIGITS Dataset

Decision Tree

	precision	recall	f1-score	support
0	1.00	0.90	0.95	50
1	0.82	0.77	0.80	53
2	0.85	0.89	0.87	56
3	0.86	0.80	0.83	60
4	0.84	0.84	0.84	62
5	0.90	0.81	0.85	53
6	0.91	0.91	0.91	47
7	0.92	0.89	0.90	61
8	0.73	0.78	0.75	49
9	0.66	0.84	0.74	49
avg / total	0.85	0.84	0.84	540

Confusion matrix:

```
[[45  0  2  0  0  2  0  0  1  0]
 [ 0 41  3  1  1  0  1  2  4  0]
 [ 0  0 50  2  1  0  1  1  1  0]
 [ 0  1  0 48  0  0  0  1  1  9]
 [ 0  1  0  0 52  1  1  1  3  3]
 [ 0  1  1  1  1 43  1  0  2  3]
 [ 0  0  1  0  2  1 43  0  0  0]
 [ 0  0  0  1  3  0  0 54  1  2]
 [ 0  3  2  1  1  0  0  0 38  4]
 [ 0  3  0  2  1  1  0  0  1 41]]
```

3^{rd} Deg Poly SVM

	precision	recall	f1-score	support
0	1.00	1.00	1.00	52
1	0.99	1.00	0.99	70
2	1.00	1.00	1.00	53
3	0.98	1.00	0.99	49
4	1.00	1.00	1.00	57
5	0.97	0.97	0.97	58
6	1.00	0.98	0.99	57
7	1.00	1.00	1.00	46
8	0.98	0.96	0.97	49
9	0.94	0.94	0.94	49
avg / total	0.99	0.99	0.99	540

Confusion matrix:

```
[[52  0  0  0  0  0  0  0  0  0]
 [ 0 70  0  0  0  0  0  0  0  0]
 [ 0  0 53  0  0  0  0  0  0  0]
 [ 0  0  0 49  0  0  0  0  0  0]
 [ 0  0  0  0 57  0  0  0  0  0]
 [ 0  0  0  0  0 56  0  0  0  2]
 [ 0  0  0  0  0  1 56  0  0  0]
 [ 0  0  0  0  0  0  0 46  0  0]
 [ 0  1  0  0  0  0  0  0 47  1]
 [ 0  0  0  1  0  1  0  0  1 46]]
```

References

- http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html
- http://scikit-learn.org/stable/modules/svm.html