HW2

April 24, 2015

```
In [1]: %matplotlib inline
        import sklearn
        import numpy as np
        import matplotlib.pyplot as plt
        plt.style.use('ggplot')
```

0.1 2d Simulation

The problem requires building a classifier to model the XOR operation

1. We first creat a 2-d random dataset using numpy.random by picking at random from a normal distribution with mean 0 and variance 1
2. The Y labels can then be created by taking the xor of (x1>0, x2>0)
3. Build a classifier using the degree 2 polynomial kernel

- A polynomial kernel of degree 2 is given by

$$k(x_i, x_j) = (x_i^T x_j)^2 = (x_{i1} * x_{j1} + x_{i2} * x_{j2})^2 = (x_{i1} * x_{j1})^2 + (x_{i2} * x_{j2})^2 + 2x_{i1} * x_{j1} * x_{j2} * x_{i2}$$

therefore $\phi(x) = x_1^2 + x_2^2 + \sqrt{2}x_1 * x_2$ hence for points where both $(x_1 < 0, x_2 < 0) or (x_1 > 0, x_2 > 0)$, we have $\phi(x) > 0$ and for points where either one is less than zero $x_1 < 0 or x_2 < 0$ but not both, $\phi(x) < 0$
Thus clearly separating the two classes. For this reason the polynomial kernel is chosen

```
In [2]: from sklearn.svm import SVC
        np.random.seed(0)
        X = np.random.randn(100,2)
        Y = np.logical_xor(X[:,0] > 0, X[:, 1] > 0)
        clf = SVC(kernel="poly", degree=2)
        clf.fit(X, Y)
        print "Number of support vectors-%s" % clf.support_vectors_.shape[0]
        print "Number of support vectors of class0 is %s and for class1 it is %s" % (clf.n_support_[0],
                                                                                     clf.n_support_[1])

        w = clf.dual_coef_
        w_norm = np.sqrt(np.sum(w**2))
        margin=1.0/w_norm
        print "Margin is %s" % margin
```

```
Number of support vectors-43
Number of support vectors of class0 is 21 and for class1 it is 22
Margin is 0.155220463955
```

Now in order to calculate the margin, we use the support vectors coefficients given by the dual_coef_ and use the formula margin=(1/||W||)

Removing Non-Support vector points

```
In [3]: X_without_nonsupport = X[clf.support_]
        print X_without_nonsupport.shape
        clf_n = SVC(kernel="poly", degree=2).fit(X_without_nonsupport, Y[clf.support_])
        print "Number of support vectors = %s" % clf_n.support_vectors_.shape[0]
        w = clf_n.dual_coef_
        w_norm = np.sqrt(np.sum(w**2))
        margin=1.0/w_norm
        print "Margin is %s" % margin
```

```
(43, 2)
Number of support vectors = 43
Margin is 0.155220463955
```

From the above, we can see that the Margin does not change when non support vector points are deleted

Removing Support Vector points

```
In [4]: supportvector_indices = clf.support_

        mask = np.ones(X.shape[0], np.bool)
        mask[clf.support_] = 0
        X_without_support = X[mask]
        clf_n = SVC(kernel="poly", degree=2).fit(X_without_support, Y[mask])

        print "Number of support vectors = %s" % clf_n.support_vectors_.shape[0]

        w = clf_n.dual_coef_
        w_norm = np.sqrt(np.sum(w**2))
        margin=1.0/w_norm
        print "Margin is %s" % margin
```

```
Number of support vectors = 10
Margin is 0.316227766017
```

*** We can see that, by deleting support vectors the margin has increased ***

0.2 3-D simulation

```
In [5]: from sklearn.svm import SVC
        X3 = np.random.randn(200,3)
        Y3 = np.logical_xor(X3[:,0] > 0, X3[:, 1] > 0, X3[:, 2] >0)
        clf = SVC(kernel="poly", degree=2)
        clf.fit(X3, Y3)
        print "Number of support vectors-%s" % clf.support_vectors_.shape[0]
        print "Number of support vectors of class0 is %s and for class1 it is %s" % (clf.n_support_[0],
                                                                                     clf.n_support_[1])
        w = clf.dual_coef_
        w_norm = np.sqrt(np.sum(w**2))
        margin=1.0/w_norm
        print "Margin is %s" % margin
```

```
Number of support vectors-102
Number of support vectors of class0 is 51 and for class1 it is 51
Margin is 0.100562379172
```

Removing non support vectors

```
In [6]: X_without_nonsupport = X3[clf.support_]
        clf_n = SVC(kernel="poly", degree=2).fit(X_without_nonsupport, Y3[clf.support_])
        print "Number of support vectors = %s" % clf_n.support_vectors_.shape[0]
        w = clf_n.dual_coef_
        w_norm = np.sqrt(np.sum(w**2))
        margin=1.0/w_norm
        print "Margin is %s" % margin
```

```
Number of support vectors = 102
Margin is 0.100560288208
```

Removing support vectors

```
In [7]: supportvector_indices = clf.support_

        mask = np.ones(X3.shape[0], np.bool)
        mask[clf.support_] = 0
        X_without_support = X3[mask]
        clf_n = SVC(kernel="poly", degree=2).fit(X_without_support, Y3[mask])

        print "Number of support vectors = %s" % clf_n.support_vectors_.shape[0]

        w = clf_n.dual_coef_
        w_norm = np.sqrt(np.sum(w**2))
        margin=1.0/w_norm
        print "Margin is %s" % margin
```

```
Number of support vectors = 20
Margin is 0.257332691658
```

Again with a higher dimensional dataset we can see that the margin does not change when only non-support vector points are deleted. Whereas it increases when support vector points are deleted

```
In [ ]:
```