mlass: Machine Learning Algorithms

Guangchuang Yu

Jinan University, Guangzhou, China

April 11, 2012

1 Linear Regression with one variable

```
> data(ex1data1)
> theta <- c(0,0)
> linReg <- gradDescent(X, y, theta, alpha=0.01, max.iter=1500)
> getTheta(linReg)

          [,1]     [,2]
[1,] -3.630291 1.166362
```

2 K-Means Clustering Algorithm

The K-means algorithm is a method to automatically cluster similar data examples together. Concretely, you are given a training set $\{x^{(1)}, ..., x^{(m)}\}$ (where $x^{(i)} \in \mathbb{R}^{\aleph}$), and want to group the data into a few cohesive clusters.

The intuition behind K-means is an iterative procedure that starts by guessing the initial centroids, and then refines this guess by repeatedly assigning examples to their closest centroids and then recomputing the centroids based on the assignments.

The K-means algorithm is as follows:

- Initialize centroids

  In *mlass* package, parameter *centers* can be set to K, which will initialize K centroids randomly, or user specific centroids.

- Refines the centroids

  Find closest centroids for each data point.

  Assign each data point to the closest centroid.

  Recompute the centroids based on the assignments.

  Repeat this procedure until it reach *max.iter* (default is 10).

The K-means algorithm will always converge to some final set of means for the centroids. Note that the converged solution may not always be ideal and depends on the initial setting of the centroids. Therefore, in practice the K-means algorithm is usually run a few times with different random initializations. One way to choose between these different solutions from different random initializations is to choose the one with the lowest cost function value (distortion).

Case Study:

```
> plot(linReg, xlab="Population of City in 10,000s", ylab="Profit in $10,000s")
```

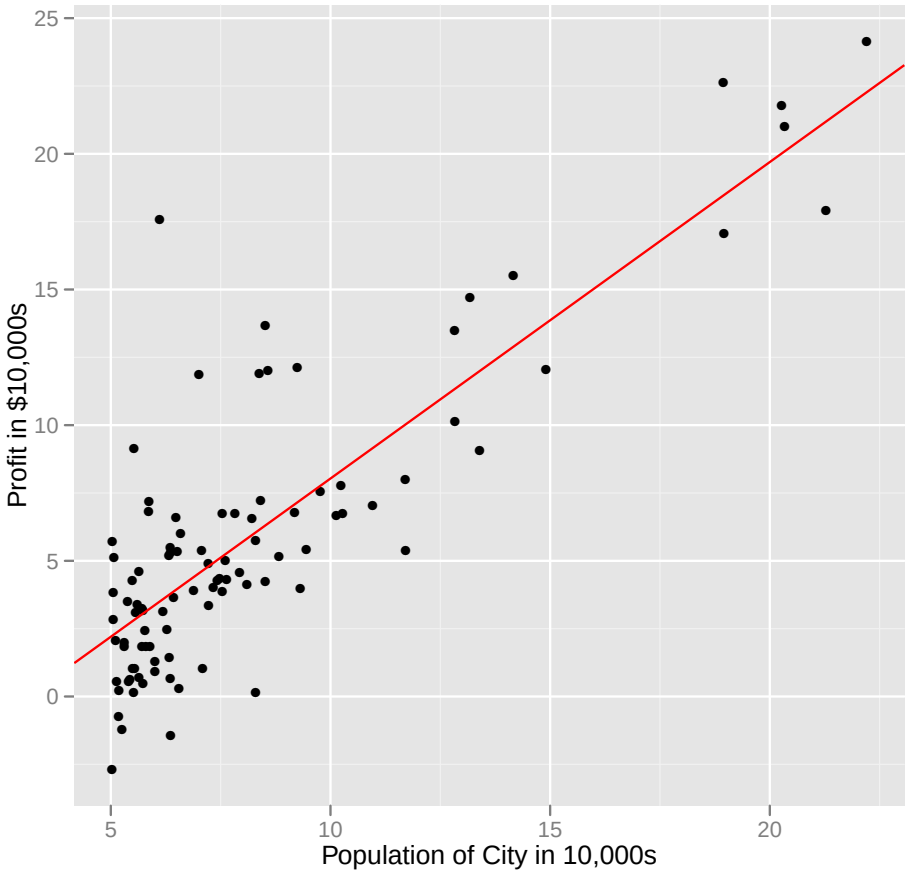

Figure 1: Linear Regression with One Variable

- Example in ML-class `http://ml-class.org`.

```
> data(ex7data2)
> initCentroids <- matrix(c(3,3,6,2,8,5), byrow=T,ncol=2)
> xx <- kMeans(X, centers=initCentroids)
> ## accessing result items
> xx["clusters"]

  [1] 1 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [28] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [55] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [82] 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[109] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[136] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[163] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[190] 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[217] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3
[244] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3
[271] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[298] 3 3 1

> xx["centroids"]

           V1       V2
[1,] 1.953995 5.025570
[2,] 3.043671 1.015410
[3,] 6.033667 3.000525
```

- IRIS dataset.

```
> data(iris)
> iris.data=as.matrix(iris[,-5])
> x=kMeans(iris.data, centers=3)
> l <- sapply(1:3, function(i)
+             names(which.max(table(iris[x["clusters"] == i,5]))))
> ## clustering accuracy
> sum(as.numeric(factor(iris[,5], levels=l)) == x["clusters"])/length(iris[,5])

[1] 0.8933333
```

The `plot` function fo visualizing the clustering result only support two features. User can use it to visualize the clustering result, with only the first two columns in iris data plotted.

K means algorithm can apply for image compression. As an example, please refer to: `http://ygc.name/2011/12/26/image-compression-using-kmeans/`.

3 Support Vector Machine

```
> data(ex6data1)
> m <- svmTrain(X,y, C=1, kernelFunction="linearKernel")
> head(m["X"])
```

```
> plot(xx, trace=TRUE, title="Iteration number 10")
```

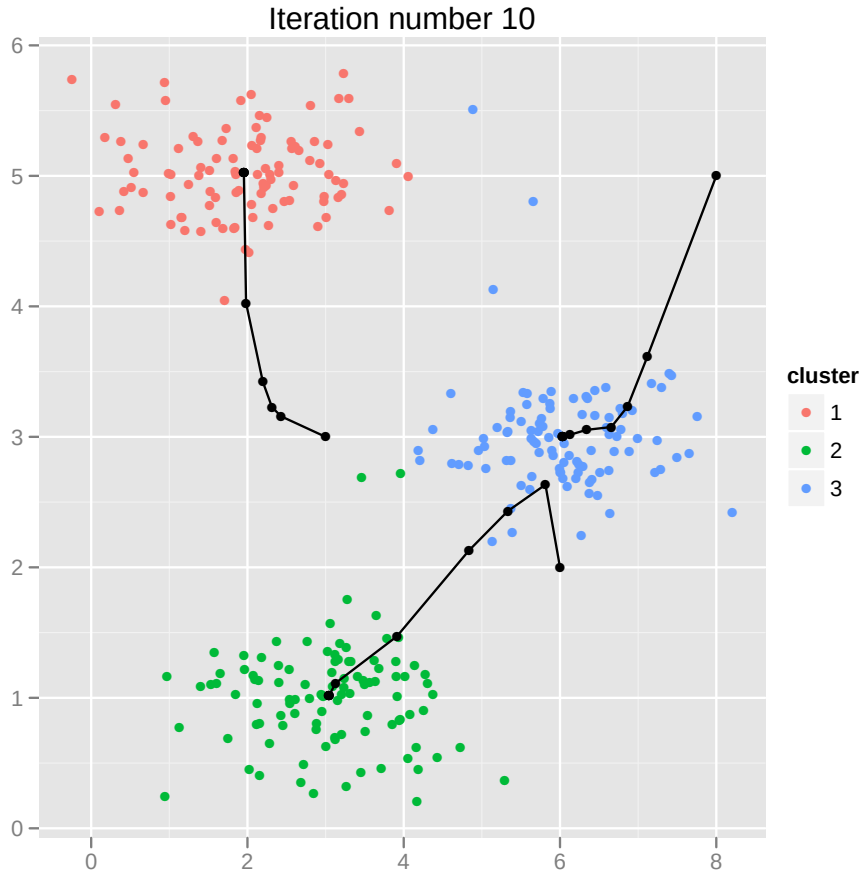

Figure 2: kMeans algorithm for clustering

```
         V1     V2
[1,] 3.5772 2.8560
[2,] 3.1048 3.0709
[3,] 1.9182 4.0534
[4,] 2.6555 3.5008
[5,] 3.0357 3.3165
[6,] 1.5841 3.3575
```

```
> head(m["y"])
```

```
[1]  1  1  1  1  1 -1
```

```
> m["w"]
```

```
          V1       V2
[1,] 1.41984 2.242872
```

```
> m["b"]
```

```
[1] -10.73196
```

```
> m["alphas"]
```

```
 [1] 1.000000e+00 1.000000e+00 7.503515e-01 1.000000e+00
 [5] 8.739368e-01 1.000000e+00 1.000000e+00 1.000000e+00
 [9] 1.000000e+00 1.387779e-17 6.242883e-01 1.000000e+00
[13] 1.000000e+00
```

```
> data(ex6data2)
> model <- svmTrain(X,y, C=1, kernelFunction="gaussianKernel")
> head(model["X"])
```

```
           V1      V2
[1,] 0.750000 0.79167
[2,] 0.761520 0.76535
[3,] 0.233870 0.86184
[4,] 0.215440 0.85892
[5,] 0.160140 0.77705
[6,] 0.058756 0.88231
```

```
> head(model["y"])
```

```
[1] 1 1 1 1 1 1
```

```
> model["w"]
```

```
             V1       V2
[1,] -0.5476186 1.557742
```

```
> model["b"]
```

```
[1] 0.4030802
```

```
> head(model["alphas"])
```

```
[1] 1.00000e+00 1.00000e+00 1.00000e+00 1.00000e+00
[5] 1.00000e+00 2.94903e-17
```

```
> plot(m, X,y, type="linear")
```

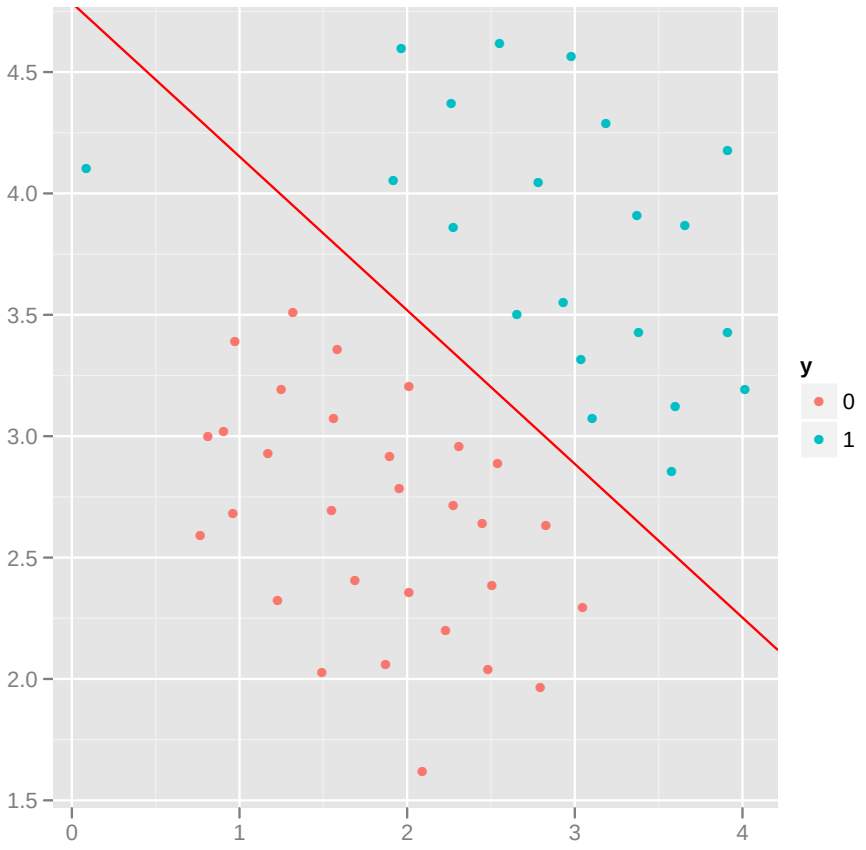

Figure 3: svm algorithm for linear decision boundaries

```
> plot(model, X,y, type="nonlinear")
```

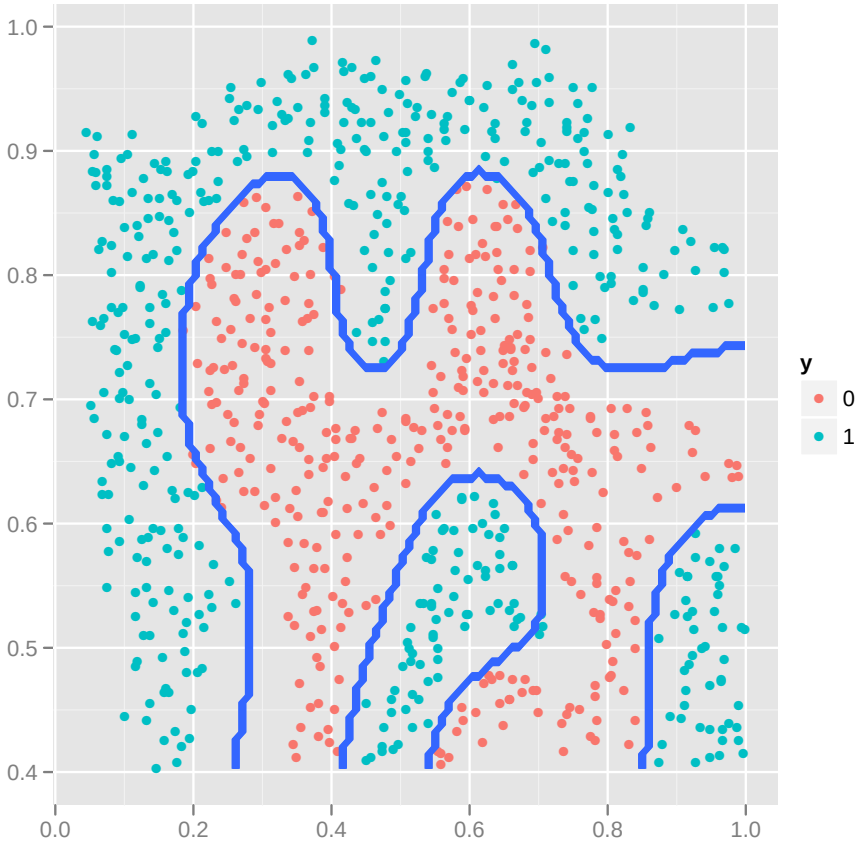

Figure 4: svm algorithm for non-linear decision boundaries

4 Session Information

The version number of R and packages loaded for generating the vignette were:

```
R version 2.15.0 (2012-03-30)
Platform: i686-pc-linux-gnu (32-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=C                 LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets
[6] methods   base

other attached packages:
[1] ggplot2_0.9.0 mlass_0.3.0

loaded via a namespace (and not attached):
 [1] MASS_7.3-17        RColorBrewer_1.0-5
 [3] colorspace_1.1-1   dichromat_1.2-4
 [5] digest_0.5.2       grid_2.15.0
 [7] memoise_0.1        munsell_0.3
 [9] plyr_1.7.1         proto_0.3-9.2
[11] reshape2_1.2.1     scales_0.2.0
[13] stats4_2.15.0      stringr_0.6
[15] tools_2.15.0
```