Announcements

- IBM Lecture on Watson Analytics will be next Monday March 07 in **RB 3201** http://carleton.ca/ims/rooms/river-building-3201/
- **Schedule of project presentations.** Enter your preferences to the file shared on Slack
- Details about **Data Day 3.0**
  - Register (free) and attend Data Day on **Tuesday March 29** http://carleton.ca/cuids/cu-events/data-day-3-0-2/
  - Consider participating in **Graduate Student Poster Competition** (prizes: 750$, 500$, 250$ for 1st, 2nd and 3rd place, respectively) http://carleton.ca/cuids/cu-events/data-day-3-0-graduate-student-poster-competition/
  - Volunteers wanted. Please email Kathryn Elliot (kathryn.elliott@carleton.ca) if interested

Machine Learning

February 29, 2016

Naïve Bayes Classification

Naive Bayes classifiers are especially useful for problems:

- with many input variables,
- categorical input variables with a very large number of possible values,
- text classification.

Naive Bayes would be a good first attempt at solving the categorization problem.

Naïve Bayes Classification

- Applicable for categorical response with categorical predictors.
- *Bayes theorem* says that

$$P(Y = y|X_1 = x_1, X_2 = x_2) = \frac{P(Y = y)P(X_1 = x, X_2 = x_2|Y = y)}{P(X_1 = x_1, X_2 = x_2)}$$

- The denominator can be expanded by conditioning on Y

$$P(X_1 = x_1, X_2 = x_2) = \sum_z P(X_1 = x_1, X_2 = x_2|Y = z)P(Y = z)$$

- The Naïve Bayes method is to assume the X_j are mutually conditionally independent, i.e.

$$P(X_1 = x_1, X_2 = x_2|Y = z) = P(X_1 = x_1|Y = z)P(X_2 = x_2|Y = z)$$

- Now the probabilities on the right-hand side can be estimated by counting from the data.

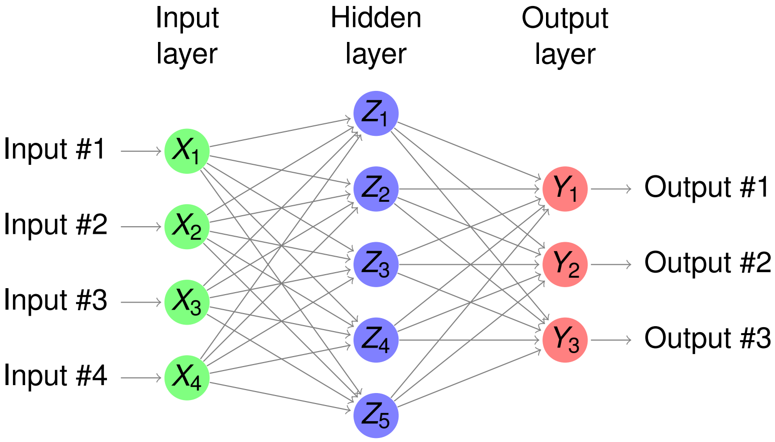
Example of Naïve Bayes

```
library(e1071)
D <- mutate(Default, income=cut(income, 3), balance=cut(balance, 2))
nb.D <- naiveBayes(default~., data=D, subset=train)
     *   *   *
A-priori probabilities:
Y
        No        Yes
0.96570645 0.03429355

Conditional probabilities:
     student
Y             No       Yes
  No   0.7073864 0.2926136
  Yes  0.6181818 0.3818182

     balance
Y      (-2.65,1.33e+03] (1.33e+03,2.66e+03]
  No         0.86454029          0.13545971
  Yes        0.09090909          0.90909091

     income
Y      (699,2.5e+04] (2.5e+04,4.93e+04] (4.93e+04,7.36e+04]
  No       0.3242510          0.5497159           0.1260331
  Yes      0.3927273          0.4836364           0.1236364
```

Example of Naïve Bayes

```
D <- mutate(Default, income=cut(income, 10), balance=cut(balance, 10))
nb.D <- naiveBayes(default~., data=D, subset=train)
nb.pred <- predict(nb.D, subset(D, test))
table(Actual=D$default[test], Predicted=nb.pred)
      Predicted
Actual   No  Yes
   No  1905   18
   Yes   40   18
```

Neural Networks



Neural Networks

$$Z_m = \sigma(\alpha_{0m} + \alpha_{1m}X_1 + \cdots \alpha pmX_p)$$

$$Y_j = \beta_{0j} + \beta_{1j}Z_1 + \cdots + \beta_{Mj}Z_M$$

- The input neurons are attached to the predictors $X_1, \ldots, X_p$.
- They are activated by a function $\sigma(v) = \frac{1}{1+e^{-v}}$.
- The neurons in the hidden layer, $Z_1, \ldots, Z_m$ are linear combinations of the inputs.
- There may be zero, one, or multiple hidden layers, with each layer being a linear combination of the previous one.
- The last layer is attached to the outputs.

Neural Networks Example

```
> library(nnet)
> nnet.fit <- nnet(default~., data=Default, subset=train, size=5)
# weights:  26
initial  value 6553.347412
iter  10 value 1136.024073
iter  20 value 1135.901203
final  value 1135.901077
converged

> summary(nnet.fit)
a 3-5-1 network with 26 weights
options were - entropy fitting
 b->h1 i1->h1 i2->h1 i3->h1
 -0.10  -0.22  -0.37  -0.47
 b->h2 i1->h2 i2->h2 i3->h2
  0.05  -0.46  -0.25   0.25
 b->h3 i1->h3 i2->h3 i3->h3
 -0.33   0.55   0.44   0.40
 b->h4 i1->h4 i2->h4 i3->h4
  0.30   0.27   0.08  -0.28
 b->h5 i1->h5 i2->h5 i3->h5
 -0.04   0.01  -0.06  -0.07
  b->o  h1->o  h2->o  h3->o  h4->o  h5->o
-22.19  -0.01   8.29  10.50   0.18   0.35
```

Neural Networks Example

```
> nnet.pred <- predict(nnet.fit, newdata=subset(Default, test),
   type="class")
> table(Actual=Default$default[test], Predicted=nnet.pred)
      Predicted
Actual   No
   No  1939
   Yes   76
```

- The table is missing the "Yes" column because the neural network didn't predict any positives.
- The neural network model is over-parametrized and there is danger of over-fitting.
- The minimization is unstable and random initialization leads to different solution each time.

K-Means Clustering

- Pick a number of clusters, say K.
- Start with a random assignment of each observation to one of the K clusters.
- For each cluster, compute the centroid as the mean of the points in the cluster.
- Reassign observations to clusters, with each observation going to the cluster with the nearest centroid.
- Repeat until convergence.

K-Means Clustering

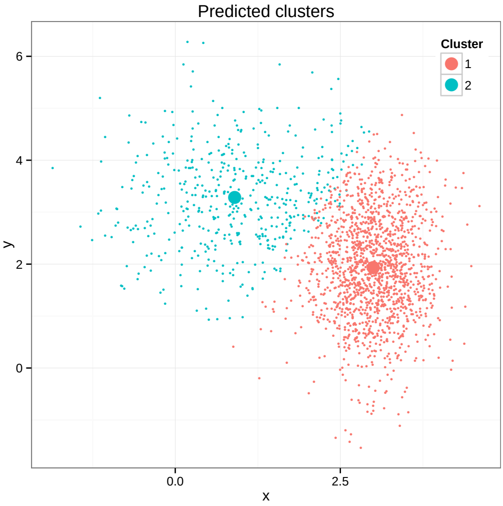
Example with simulated data.

```
pts <- read.csv('pts_2clusters.csv', header=TRUE)
qplot(x, y, data=pts, color=cl) + labs(color="Cluster")
```

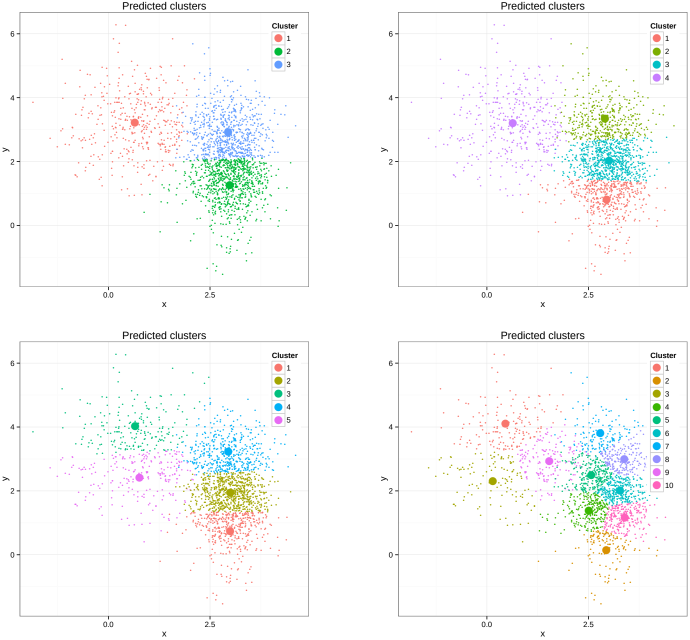
K-Means Clustering

Solve for two clusters.

```
km.out <- kmeans(pts, 2)
qplot(x, y,  data=mutate(pts, cl.1=factor(km.out$cluster)), color=cl.1)
```



K-Means Clustering



Hierarchical Clustering

- Here we don't pick the number of clusters in advance, this is decided by the algorithm.
- We need a distance or *dissimilarity* measure
- Start with each point in its own cluster.
- Compute all pairwise dissimilarities and merge the two most similar clusters.
- Repeat until some stopping criterion is reached.

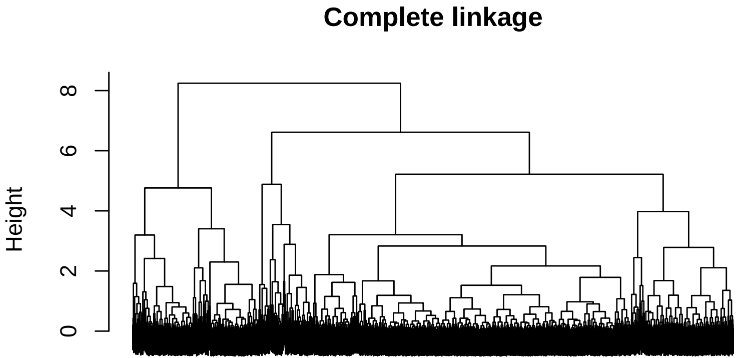
- To compute dissimilarity between two clusters, *A* and *B*, one may look at different possibilities.
  - Take the dissimilarity of the two centroids.

    Compute all pairwise dissimilarities between points in *A* and points in *B*.
  - Complete linkage: take the maximum
  - Single linkage: take the minimum;
  - Average linkage: take the average.

Hierarchical Clustering

Example with the same simulated data.

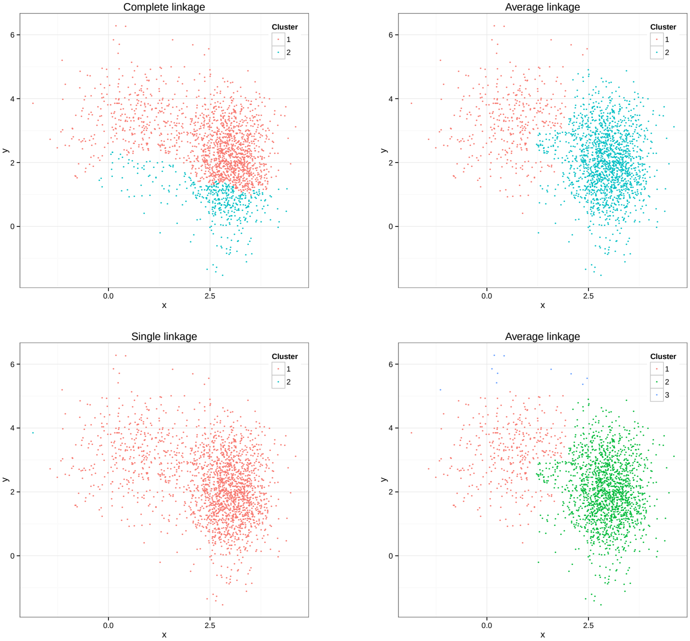
```
library(grDevices)
hc.out <- hclust(dist(pts[c('x','y')]), method="complete")
plot(hc.out, xlab="", main="Complete linkage", sub="")
```



Hierarchical Clustering

```
cl.1 <- cutree(hc.out, k=2)
qplot(x, y,  data=mutate(pts, cl.1=factor(cl.1)), color=cl.1)
plot(hc.out, xlab="", main="Complete linkage", sub="")
```

Hierarchical Clustering



Association rules

- Data is a binary matrix with columns corresponding to products and rows corresponding to baskets.
- Entry (i, j) is TRUE if customer i purchased product j.
- *Apriori* algorithm looks at most probable sets of products and combines them

Association rules

- Association rule is a claim such as: $A \,\&\, B \Rightarrow C$.
- Support for the rule is the probability of all items being together

$$\text{Support}(A \,\&\, B \,\&\, C) = \frac{\text{Number of baskets with A, B and C}}{\text{Total number of baskets}}$$

- Confidence of a rule is the conditional probability of the implied item

$$\text{Confidence}(A \,\&\, B \Rightarrow C) = \frac{\text{Support}(A \,\&\, B \,\&\, C)}{\text{Support}(A \,\&\, B)}$$

- Lift of a rule is

$$\text{Lift}(A \,\&\, B \Rightarrow C) = \frac{\text{Confidence}(A \,\&\, B \Rightarrow C)}{\text{Support}(C)}$$

Association rules

- We start by computing the supports of all single items and sort them.
- Then prune at say 80% and compute the support of all rules with two items of the remaining ones.
- Sort and prune. Then proceed with rules with three items, not including pairs that have been pruned. And so on.

Association rules

```
> mb <- read.csv('MarketBasket.csv')  # Simulated data
> library(arules)
> rules <- apriori(mb, parameter=list(supp=0.8, conf=0.8, target="rules"

Parameter specification:
 confidence minval smax arem  aval originalSupport support minlen maxlen
        0.8    0.1    1 none FALSE            TRUE     0.8      1     10
   ext
 FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[5 item(s), 500 transaction(s)] done [0.00s].
sorting and recoding items ... [3 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [12 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

Association rules

```
> inspect(rules)
   lhs       rhs   support confidence      lift
1  {}     => {V4}   0.932  0.9320000 1.0000000
2  {}     => {V1}   0.950  0.9500000 1.0000000
3  {}     => {V3}   1.000  1.0000000 1.0000000
4  {V4} => {V1}   0.882  0.9463519 0.9961599
5  {V1} => {V4}   0.882  0.9284211 0.9961599
6  {V4} => {V3}   0.932  1.0000000 1.0000000
7  {V3} => {V4}   0.932  0.9320000 1.0000000
8  {V1} => {V3}   0.950  1.0000000 1.0000000
9  {V3} => {V1}   0.950  0.9500000 1.0000000
10 {V1,
    V4} => {V3}   0.882  1.0000000 1.0000000
11 {V3,
    V4} => {V1}   0.882  0.9463519 0.9961599
12 {V1,
    V3} => {V4}   0.882  0.9284211 0.9961599
```