```
118     def similarityMethod(self, class1, class2):
119         return [ 1 if point.similarityToPoint(class1.mean) < point.similarityToPoint(c
120 
121 
122 
123 # NOTE: - covariances:  ********************************************************  cova
124 
125 def covarianceWithinClasses(class1, class2):
126     ''' within-class-covariance '''
127     sumOfClass1 = nu.array([ (row.reshape(2,1) - class1.mean) * (row.reshape(2,1) - cl
128     sumOfClass2 = nu.array([ (row.reshape(2,1) - class2.mean) * (row.reshape(2,1) - cl
129     return (sumOfClass1 + sumOfClass2) / (class1.len + class2.len)
130 
131 
132 def covarianceBetweenClasses(class1, class2):
133     ''' between-class-convariance '''
134     meanTotal = (class1.mean + class2.mean) / 2.0
135     a = class1.len * (class1.mean - meanTotal) * (class1.mean - meanTotal).T
136     b = class2.len * (class2.mean - meanTotal) * (class2.mean - meanTotal).T
137     return (a + b) / (class1.len + class2.len)
138 
139 
140 def covarianceTotal(class1, class2):
141     ''' total covariance '''
142     meanTotal = (class1.mean + class2.mean) / 2.0
143     allPoints = nu.insert(class1.toNdarray, class1.len, class2.toNdarray, axis=0)
144 
145     sum = nu.array([ (row.reshape(2,1) - meanTotal) * (row.reshape(2,1) - meanTotal).T
146     return sum / (class1.len + class2.len)
147 
148 
149 
150 # NOTE: - eigen value and vectors:  ****************************************  eige
151 
152 Eigen = namedtuple('Eigen', 'values vectors')
153 
154 def discriminantOf(class1, class2):
155     A = covarianceBetweenClasses(class1, class2)
156     B = covarianceWithinClasses(class1, class2)
157     values, vectors = la.eig(A, B)
158     vectors = nu.array([ vector.reshape(2,1) / nu.linalg.norm(vector) for vector in ve
159 
160     return Eigen(values, vectors)
161 
162 
```