Machine Learning 1

Solved Problems

1 Team members:

- Luis Sierra Muntané (luis.sierra@est.fib.upc.edu)
- Àlex Batlle Casellas (alex.batlle@est.fib.upc.edu)
- Aleix Torres i Camps (aleix.torres.camps@est.fib.upc.edu)

2 Problems

1. (a) Write a procedure to generate random samples according to a normal distribution $\mathcal{N}(\mu, \Sigma)$ in d dimensions.

   SOLUTION:

   Based on the information in Wikipedia's Multivariate Normal Distribution page, and in particular in this part, we see that a $d-$dimensional vector X is normally distributed (it is a *normal random vector*) iff there exist a vector μ, a matrix A of coefficients and a standard normal vector Z (meaning, it follows a standard normal distribution) such that $AZ + \mu = X$, with the covariance matrix $\Sigma = AA^T$. So, we will take advantage of all this situation:

   - We can calculate the matrix A by simply decomposing Σ (which is the matrix we are given) using the Cholesky decomposition method, which factors a symmetric positive-definite matrix into a product of a lower triangular matrix and its transpose. R has a built in function named `chol`, although this gives the upper triangular part of the decomposition.
   - We just have to calculate a random vector Z, and we will do so using the function `rnorm` already included in R.

   So, the solution code is this one:

```
normal = function(mu, sigma) {
    d = length(mu)
    L = t(chol(sigma))
    res = rnorm(d)
    return(L %*% res + mu)
}
```

   (b) Write a procedure to calculate the discriminant function (of the form given in Eq. 47) for a given normal distribution and prior probability $P(\omega_i)$.

   SOLUTION:

   We will suppose we are given the probability of class i, and the μ_i, Σ_i such that $p(x|\omega_i) \sim \mathcal{N}(\mu_i, \Sigma_i)$. Then, the implementation is quite straight forward given the formula in Eq. 47:

```
calc_discr = function(x, p, mu, sigma) {
  xnorm = x - mu
  inv = solve(sigma)
  d = length(mu)
  dt = det(sigma)
  return(-0.5*t(xnorm)%*%inv%*%xnorm-d/2*log(2*pi)-0.5*log(dt)+log(p))
}
```

We are using the function `solve`, which R has built-in, that solves linear systems given A, b such that $Ax = b$. If b is absent, the function returns the inverse of a matrix.

(c) Write a procedure to calculate the Euclidean distance between two arbitrary points.
SOLUTION:

```
# pre: dimensions of x and y are the same
euc = function(x,y) {
  n = length(x)
  sum = 0
  for (i in 1:n) {
    sum = sum + (x[i]-y[i])^2
  }
  return(sqrt(sum))
}
```

Note that we could have also used the `dist` function included in R, by row-binding the coordinates of x and y into a matrix, like this:

```
# pre: dimensions of x and y are the same
euc2 = function(x,y) {
  dd = rbind(x,y)
  return(dist(dd))
}
```

(d) Write a procedure to calculate the Mahalanobis distance between the mean μ and an arbitrary point x, given the covariance matrix.
SOLUTION:
We will be using the matrix formula for the Mahalanobis distance, which is the square root of a quadratic form, namely

$$\delta_M(x, y) = \sqrt{(x-y)^T \Sigma^{-1} (x-y)}.$$

```
mah = function(x, mu, sigma) {
    xnorm = x - mu
    return(sqrt(t(xnorm)%*%solve(sigma)%*%xnorm))
}
```