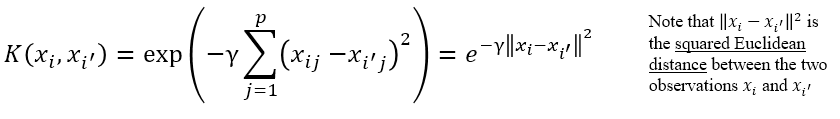
# Support Vector Machines

* Support vector machines are capable of deriving non-linear decision boundaries. They are an extension of the support vector classifier, and achieve theit non-linear solutions by means of a *kernel*.
* A kernel is a function that quantifies the similarity of two observations (just as the inner product does in a support vector classifier (see slides 7 through 10 in the PowerPoint slides for this topic for more about this statement).
* For example, a polynomial kernel replaces the linear kernel  
  with the following:
* Here, is a training constant that controls the bias-variance tradeoff and must be found, typically by using cross validation.
* Another example: a radial kernel replaces the linear kernel above with   
  
* Here, is a training constant that controls the bias-variance tradeoff which must be found.
* When evaluating the sign of an observation x, the fewer support vectors that exert an influence on the higher the model variance will be (and the lower the bias will be).
* For a point *x* that is distant from a support vector , a small value of g will make smaller, the therefore make the kernel function larger, meaning that the distant support vector will exert more influence on the classification of the training example than if g were large.
* So small g means more support vectors influence the sign of , and therefore the model will have lower variance but higher bias. Conversely, is g is large, the model will exhibit higher variance but will introduce less bias.
* So
  + Small *g* means large influence of distant support vectors
    - * which means more support vectors participate…
      * which means lower variance and higher bias
  + Large *g* means small influence of distant support vectors
    - * which means fewer support vectors participate…
      * which means higher variance and lower bias
* The Gaussian kernel is defined as follows:
* Note the similarity to the radial kernel – we have simply replaced the training constant with, both of which must be set by the modeler.
* Thus, while a large *g* in the radial kernel risks high variance, the opposite is true for the role of *s* in the Gaussian kernel:
* Which Kernel Should One Use?
  + When variance is expected to be a major consideration, start with a linear kernel
    - For example, when *p* is relatively large and n is relatively small
  + Otherwise, a low-bias (and therefore highly flexible) approach usually produces better outcomes, which suggests a polynomial, radial or Gaussian kernel may work best
* The concept of separating hyperplanes upon which SVMs are based does not lend itself naturally to more than two classes. However, one can accommodate more than two classes using a *one-versus-one* or *one-versus-all* approach. See slides 25 and 26 for more, and see the “SVM with Multiple Classes” section of the script that accompanies the Support Vector Machine lecture for an example in R.
* **R (just the important stuff)  
  rm(list=ls())**

**require(e1071)**

**set.seed(5082)**

**x <- matrix(rnorm(1000 \* 2), ncol=2)**

**x[1:333,] <- x[1:333, ]+2**

**x[334:666,] <- x[334:666, ]-2**

**y <- c(rep(-1, 666),rep(1, 334))**

**dat <- data.frame(x=x, y=as.factor(y))**

**plot(x[, 2], x[, 1], col=y+3)**

**train <- sample(1000, 800)**

**set.seed(1)**

**tune.out <- tune(svm, y~., data=dat[train,], kernel="radial",**

**ranges=list(cost=c(0.001, 0.01, 0.1, 1, 10),**

**gamma=c(0.5, 1, 2, 3, 4)))**

**summary(tune.out)**

**bestmod <- tune.out$best.model**

**summary(bestmod)**

**plot(bestmod, dat[train,])**

**pred.y <- predict(tune.out$best.model, newdata=dat[-train,])**

**(my.table <- table(actuals=dat[-train, ]$y, predictions=pred.y))**

**print(paste("The parameters of the best model are cost = ",**

**bestmod$cost, " and gamma = ",**

**bestmod$gamma, sep=''))**

**print(paste("The test error rate for this support vector machine is",**

**mean(dat[-train, "y"] != pred.y)))**