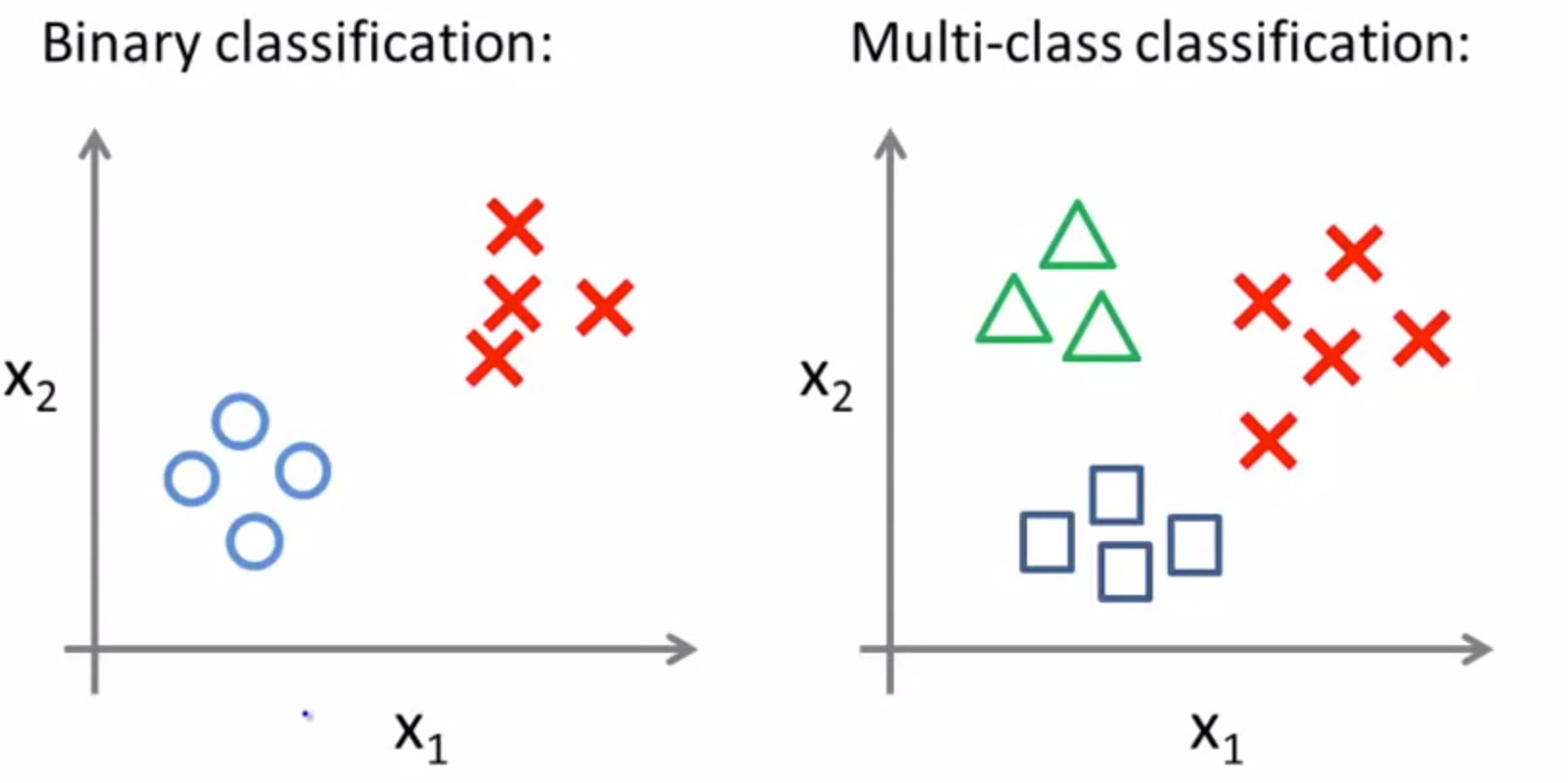
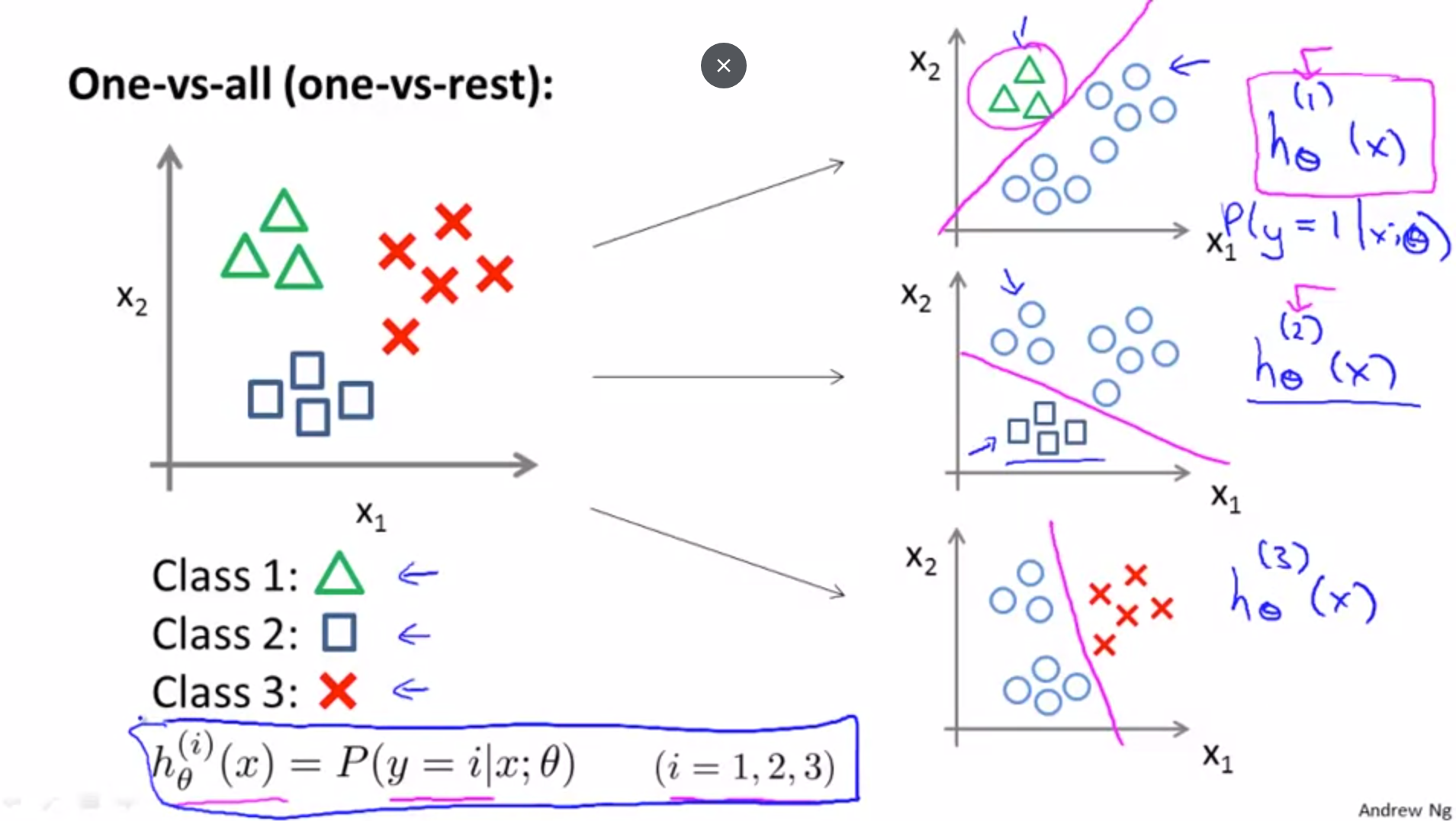
* Multi-class classification
  + E.g. email folder/tagging: y = 0 (work), 1 (friends), 2 (family), …

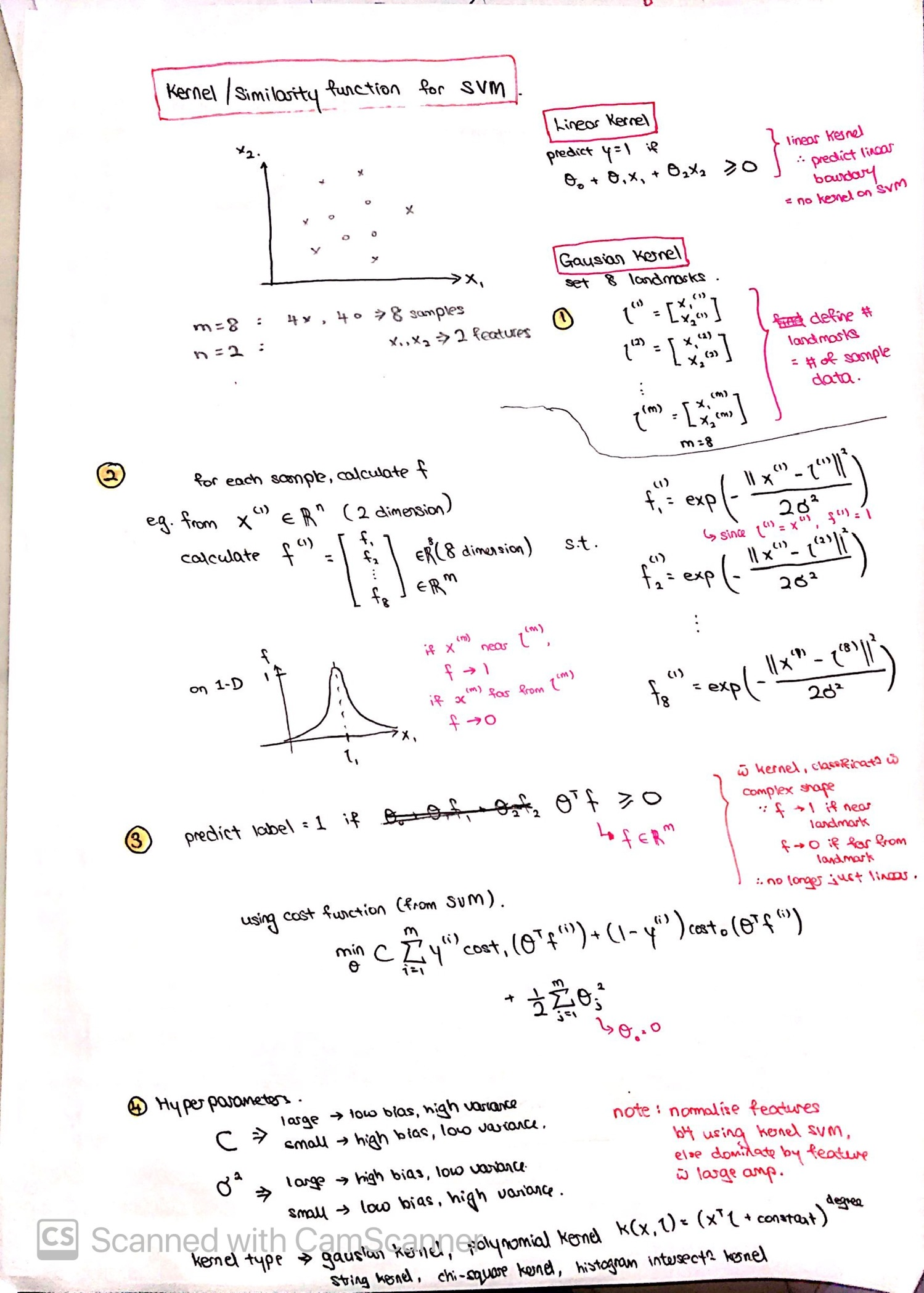


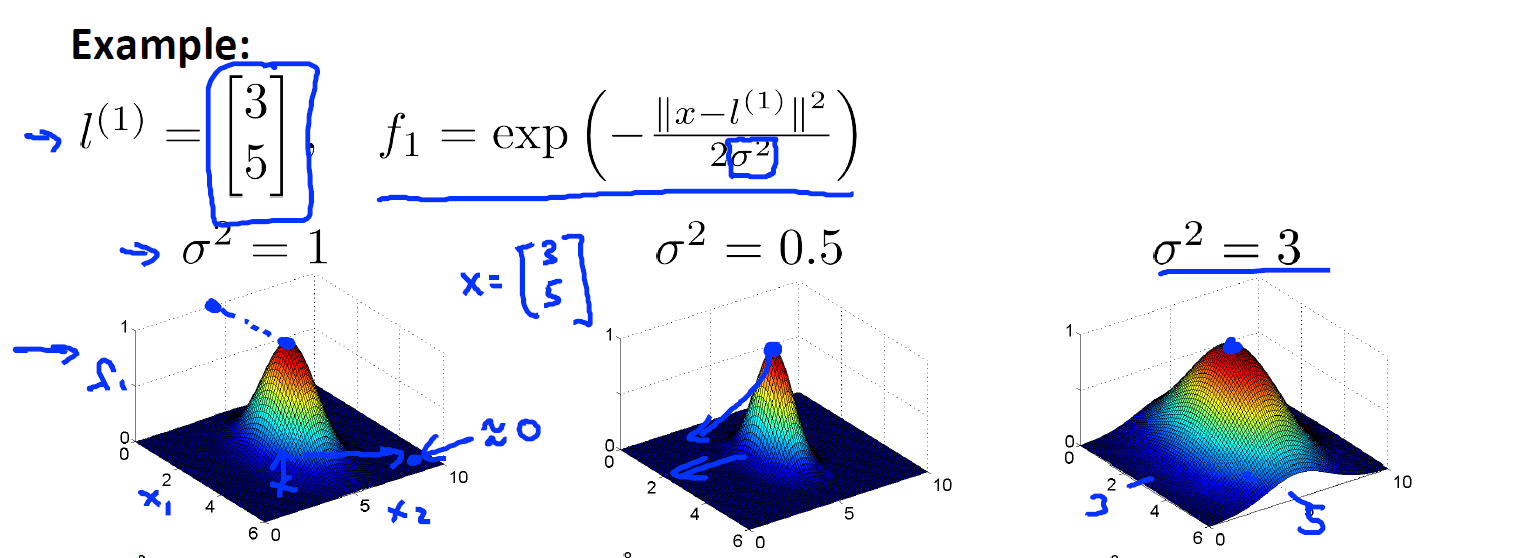
* + How: One-vs-all (one-vs-rest) method
    - Then find , (i.e. find of the 3 classifier, the label that has the highest probability



**SVM (Support Vector Machine)**





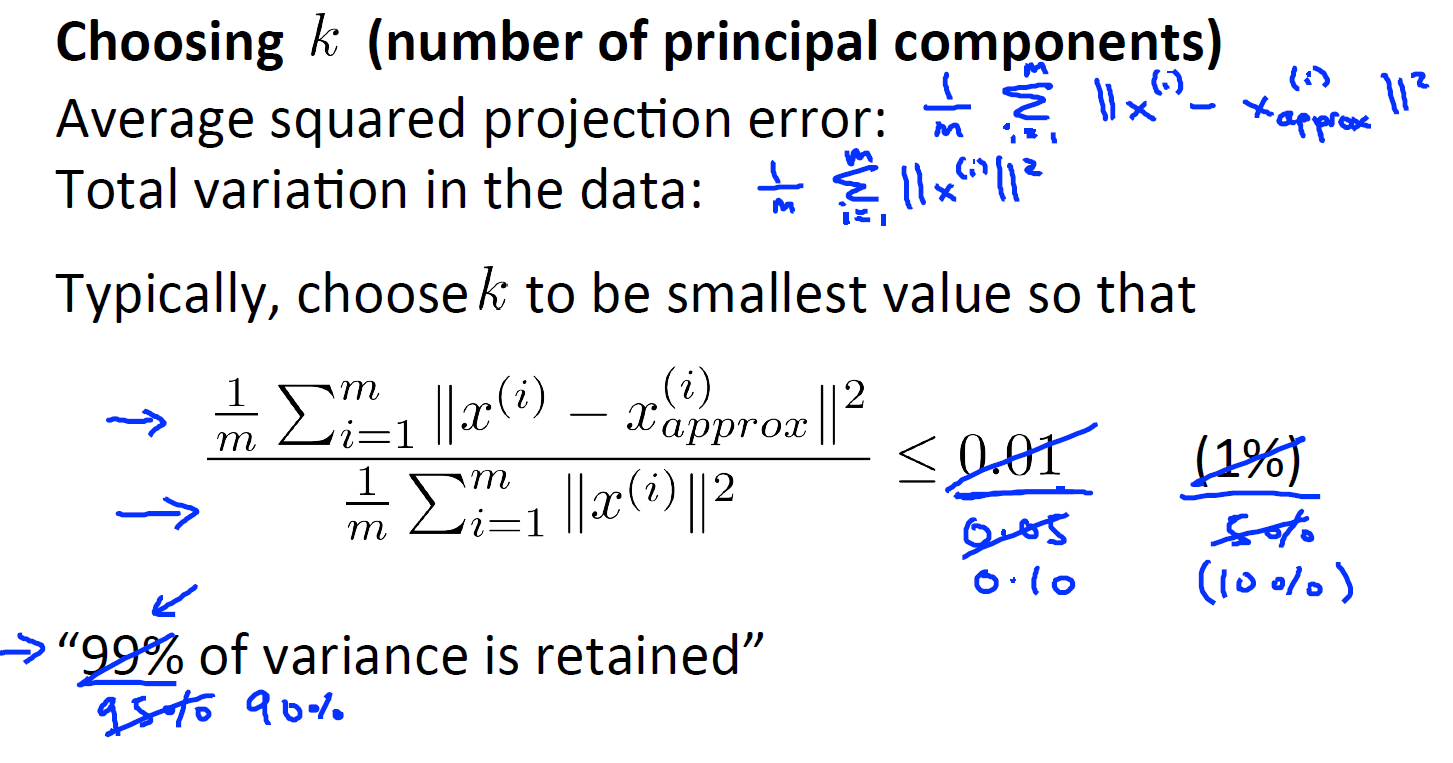


* + Can also do multi-class classification using one-vs-all method
    - i.e. make k SVM for k classes
    - Find largest class
  + “SVM (with linear kernel) or logistic regression” Vs “SVM (w gaussian kernel)
    - N: number of features, M = number of data
    - N large, M small 🡺 SVM (with linear kernel) or logistic regression
    - N small, M intermediate 🡺 SVM (w gaussian kernel)
    - N small, M large 🡺 Create/add more features, then SVM (with linear kernel) or logistic regression
    - Deep neural network works for all, especially when carefully tuned
    - N small : up to 1000
    - N large : e.g. 10,000
    - M small: less than 1000
    - M intermediate: up to 10,000
    - M large: > 50,000

**PCA (Principal Component Analysis)**

See notebook in E:\GitHub\PCA

* What is it
  + Compress dataset with n features to k features (k < n)
  + To achieve:
    - Data compression (save memory space, speed up learning)
    - Visualization (if n can be compressed to k = 2/3 for plotting)
* How
  + Step 1: Mean normalization or Feature Scaling (use if magnitude of features differs greatly)
    - Mean normalization
      * (i.e. find mean of each feature)
      * Replace xj(i) with xj(i) - µj (so that all features have 0 mean)
    - Feature Scaling
      * Replace xj(i)  with (i.e. normalize the data w mean and S.D)
      * Note: Apply only either Step 1 or Step 2, no need both
  + Step 2: Compute Covariance Matrix
    - (Sigma = Covariance Matrix)
      * Shape x(i) = (n, 1)
      * Shape of Sigma = (n, n)
  + Step 3: Compute eigenvectors of Sigma
    - [U, S, V] = svd (Sigma)
    - SVD = Singular Value Decomposition
    - U : Unitary array, gives the list of output vector direction. Shape (n, n)
    - S : Provides a mean to determine value of k to use
  + Step 4: Get new feature vectors and corresponding Z in k dimension
    - Select a value of k to use
    - From U, select the first k columns to get U\_reduce. U\_reduce dim = (n, k)
    - Get Z (dimension = (k, m))
      * Z = U\_reduce.T x X
  + Step 5 (Optional): Reconstruct back the approximate value of X from Z
    - X\_approx = U\_reduce x Z
    - Dimension = (n, m)
  + Step 6 (Optional): Calculate the % variance retained
    - Method 1:



* + - Method 2:

