Learning Vector Quantization and K-Nearest Neighbor

Jia Li

Department of Statistics
The Pennsylvania State University

Email: jiali@stat.psu.edu
http://www.stat.psu.edu/∼jiali

Learning Vector Quantization

- Developed by Kohonen. A package with document is available at: `http://www.cis.hut.fi/nnrc/nnrc-programs.html` .
- When position the prototypes, use information given by class labels, as a contrast to k-means which selects prototypes without using class labels.
- Often works better than k-means.
- The idea is to move a prototype close to training samples in its class, and move away from samples with different classes.

K-means - 5 Prototypes per Class

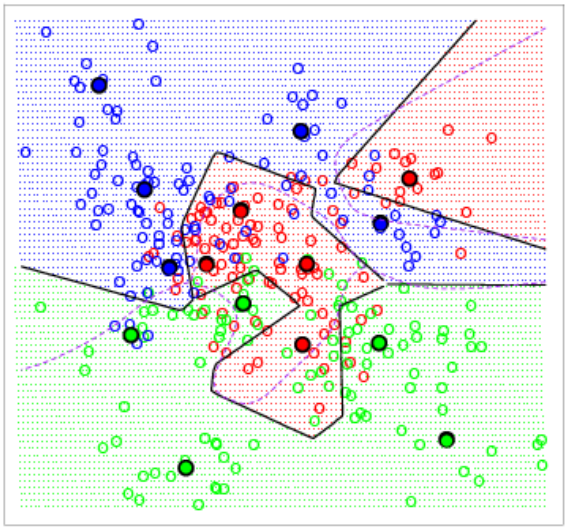
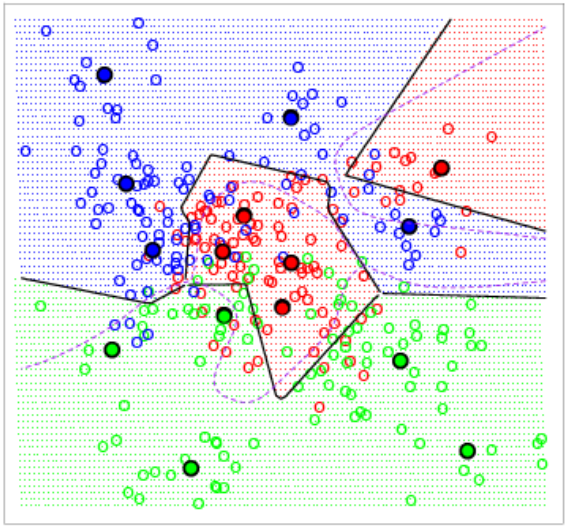


Figure 13.1: *Simulated example with three classes and five prototypes per class. The data in each class are generated from a mixture of Gaussians. In the upper panel, the prototypes were found by applying the K-means clustering algorithm separately in each class. In the lower panel, the LVQ algorithm (starting from the K-means solution) moves the prototypes away from the decision boundary. The broken purple curve in the background is the Bayes decision boundary.*

LVQ - 5 Prototypes per Class



The Algorithm

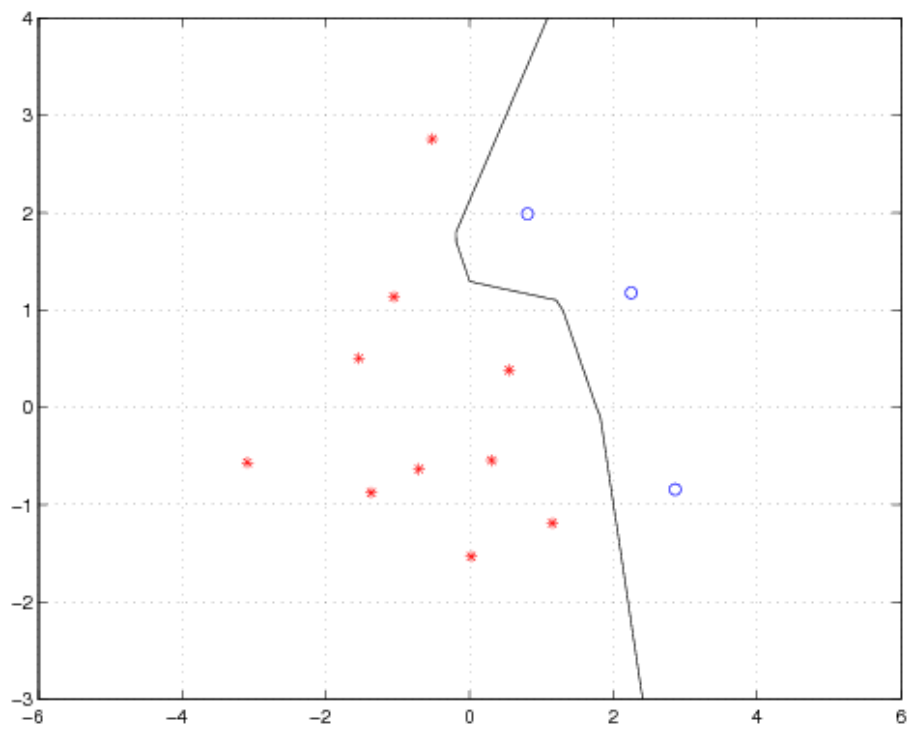
1. Start from a set of initial prototypes with classes assigned. Denote the M prototypes by $\mathcal{Z} = \{z_1, ..., z_M\}$ and their associated classes by $C(z_m)$, $m = 1, 2, ..., M$.
   - The initial prototypes can be provided by k-means.
2. Sweep through the training samples and update z_m after visiting each sample.
   - Suppose x_i is assigned to the mth prototype z_m by the nearest neighbor rule:
     $\| x_i - z_m \| \leq \| x_i - z_{m'} \|, \forall m' \neq m, 1 \leq m' \leq M$
   - If $g_i = C(z_m)$, move z_m towards the training sample:
     $z_m \leftarrow z_m + \epsilon(x_i - z_m)$ where ϵ is the *learning rate*.
   - If $g_i \neq C(z_m)$, move z_m away from the training sample:
     $z_m \leftarrow z_m - \epsilon(x_i - z_m)$
3. Step 2 can be repeated a number of times.

Experiments

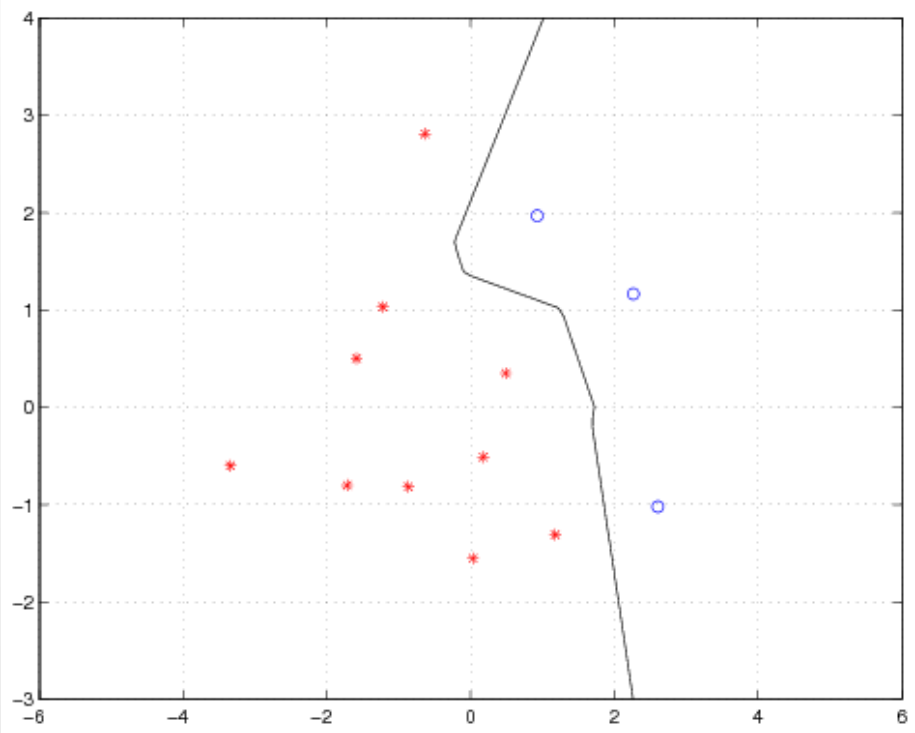
- Use the diabetes data set.
- Use prototypes obtained by k-means as initial prototypes.
- Use LVQ with $\epsilon = 0.1$.
- Results obtained after 1, 2, and 5 passes are shown below.
- Classification is not guaranteed to improve after adjusting prototypes.
- One pass with a small ϵ usually helps. But don't over do it.

Comments:

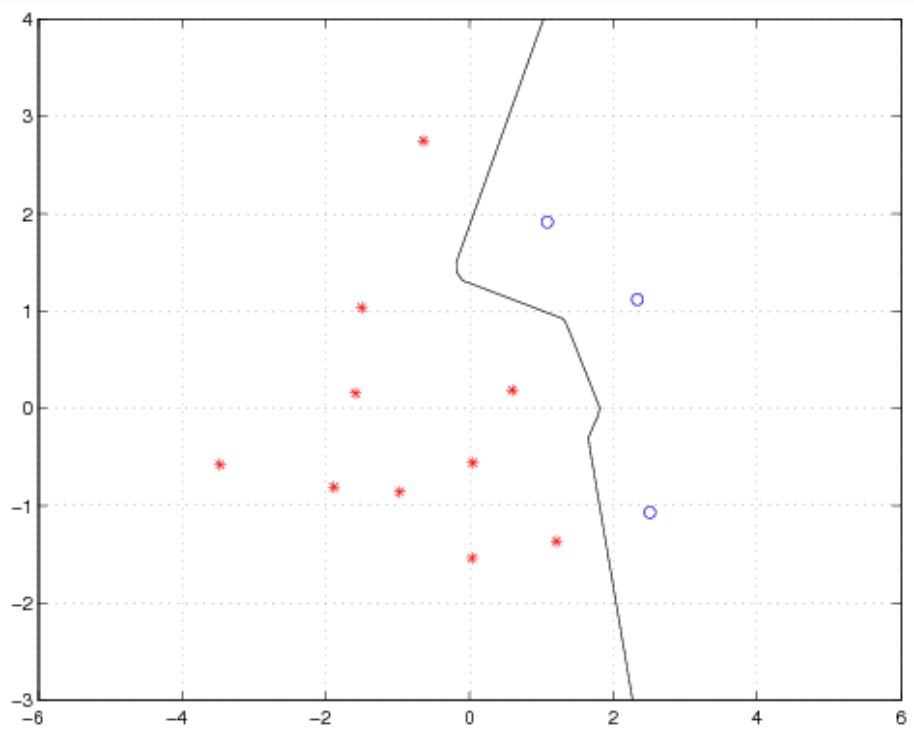
- Fine tuning often helps:
  - Select initial prototypes.
  - Adjust learning rate ϵ.
- Read the package documents for details.



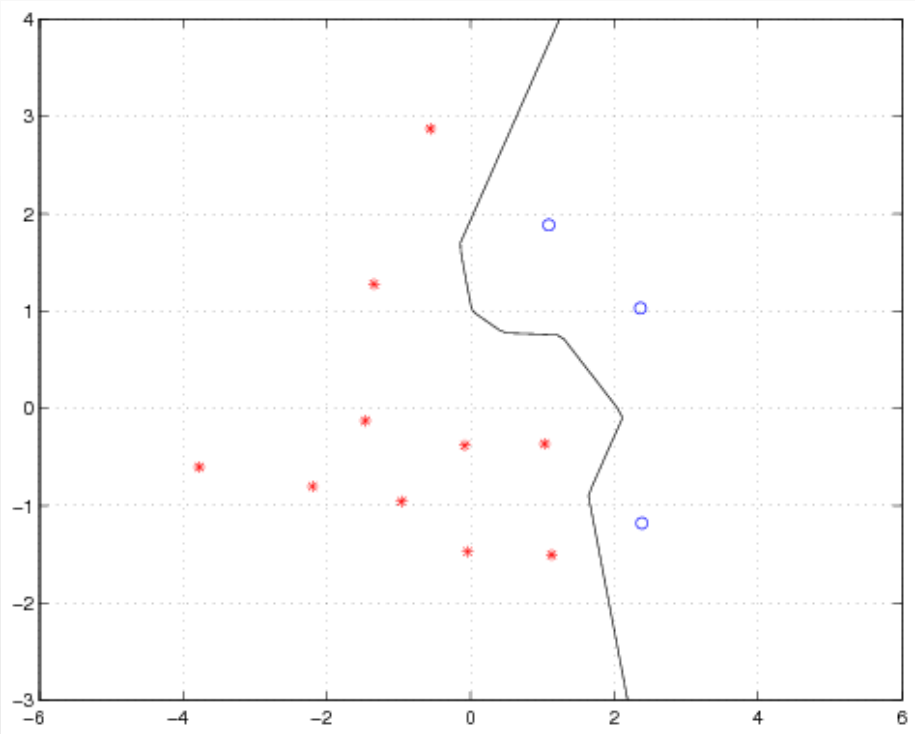
Error rate: 27.74%



Error rate: 27.61%



Error rate: 27.86%



Error rate: 32.37%

K-Nearest Neighbor Classifiers

- Given a query point x_0, find the k training samples $x_{(r)}$, $r = 1, ..., k$ closest in distance to x_0, and then classify using majority vote among the k neighbors.
- Feature normalization is often performed in pre-processing.
- Classification boundaries become smoother with larger k.

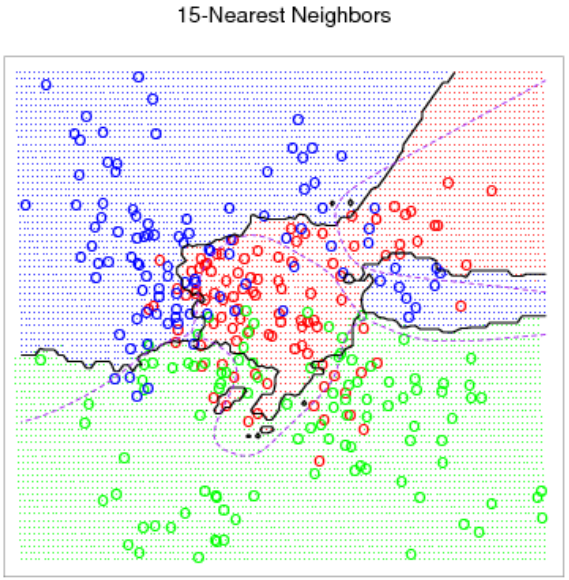
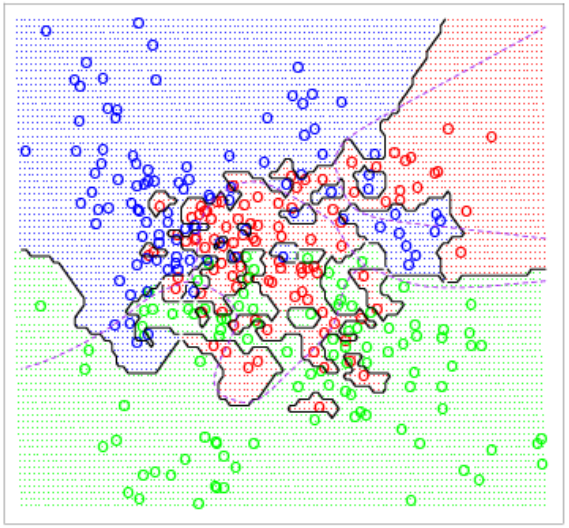


Figure 13.3: *k-nearest-neighbor classifiers applied to the simulation data of Figure 13.1. The broken purple curve in the background is the Bayes decision boundary.*

1-Nearest Neighbor



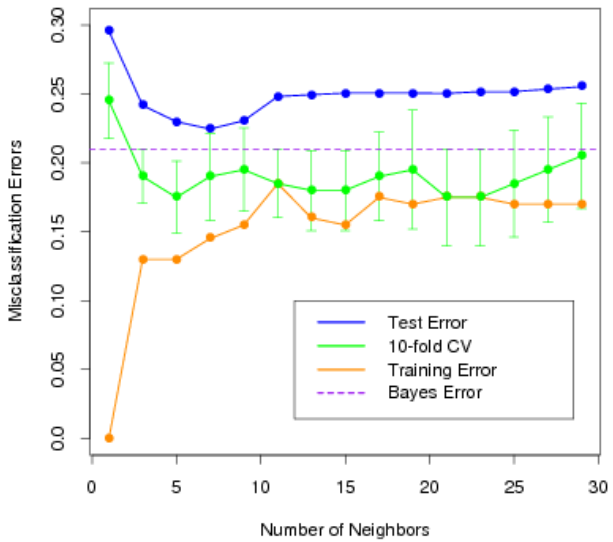
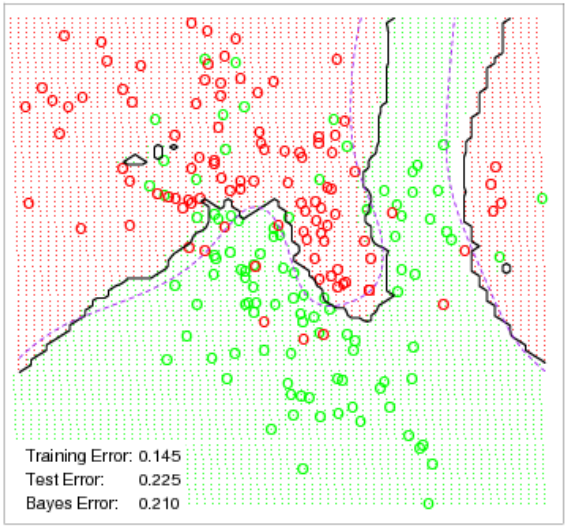


Figure 13.4: *k-nearest-neighbors on the two-class mixture data. The upper panel shows the misclassification errors as a function of neighborhood size. Standard error bars are included for 10-fold cross validation. The lower panel shows the decision boundary for 7-nearest-neighbors, which appears to be optimal for minimizing test error. The broken purple curve in the background is the Bayes decision boundary.*

7-Nearest Neighbors



A Comparative Study (ElemStatLearn)

- Two simulated problems.
- There are 10 independent features X_j, each uniformly distributed on $[0, 1]$.
- The two-class 0-1 target variable is defined as follows:

problem 1: "easy"

$$Y = I(X_1 > \frac{1}{2});$$

problem 2: "difficult"

$$Y = I\left(sign\left\{\prod_{j=1}^{3}(X_j - \frac{1}{2})\right\} > 0\right).$$

- For problem 1 (2), except X_1 (and X_2, X_3), all the other features are "noise".
- The Bayes error rates are zero.
- In each run, 100 samples used in training and 1000 used in testing.

- The figure shows the mean and standard deviation of the misclassification error for nearest-neighbors, K-means and LVQ over 10 realizations (10 simulated data sets), as the tuning parameters change.
- K-means and LVQ give almost identical results.
- For the first problem, K-means and LVQ outperform nearest neighbors, assuming the best choice of tuning parameters for each. For the second problem, they perform similarly.
- The optimal k for the k-nearest neighbor classification differs significantly for the two problems.

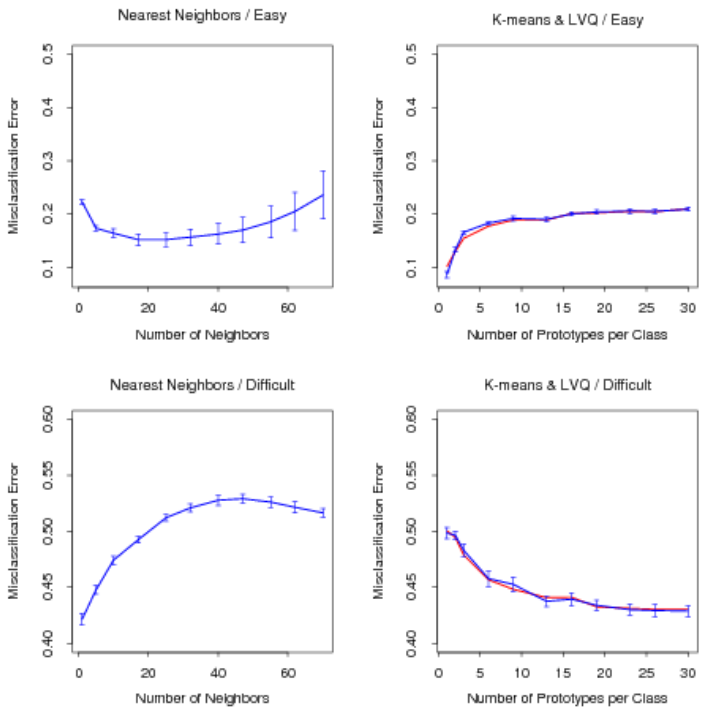


Figure 13.5: *Mean ± one standard error of misclassification error for nearest-neighbors, K-means (blue) and LVQ (red) over ten realizations for two simulated problems: "easy" and "difficult," described in the text.*

Adaptive Nearest-Neighbor Methods

- When dimension is high, data become relatively sparse.
- Implicit in nearest-neighbor classification is the assumption that the class probabilities are roughly constant in the neighborhood, and hence simple averages give good estimates.
- In high dimensional space, the neighborhood represented by the few nearest samples may not be local.
  - Consider N data points uniformly distributed in the unit cube $[-\frac{1}{2}, \frac{1}{2}]^p$. Let R be the radius of a 1-nearest-neighborhood centered at the origin. Then

$$\text{median}(R) = v_p^{-1/p} \left(1 - \frac{1}{2}^{1/N}\right)^{1/p},$$

    where $v_p r^p$ is the volume of the sphere of radius r in p dimensions.
  - The median radius quickly approaches 0.5, the distance to the edge of the cube, when dimension increases.

- Adjust distance metric locally, so that the resulting neighborhoods stretch out in directions for which the class probabilities don't change much.

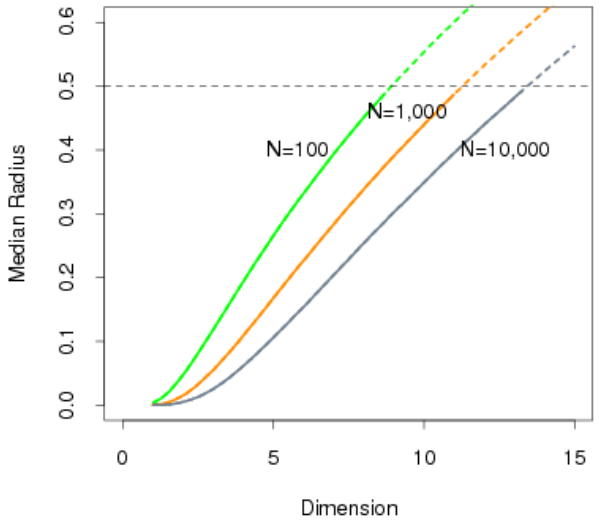


Figure 13.12: *Median radius of a 1-nearest-neighborhood, for uniform data with N observations in p dimensions.*

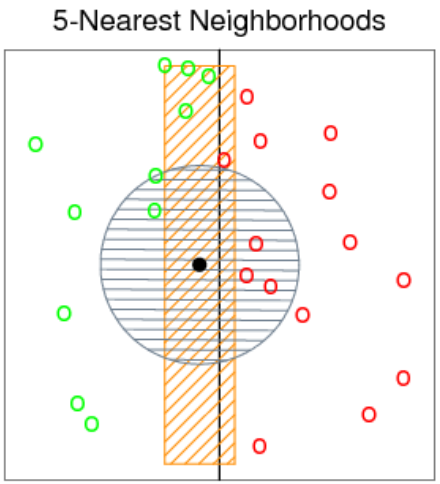


Figure 13.13: *The points are uniform in the cube, with the vertical line separating class red and green. The vertical strip denotes the 5-nearest-neighbor region using only the horizontal coordinate to find the nearest-neighbors for the target point (solid dot). The sphere shows the 5-nearest-neighbor region using both coordinates, and we see in this case it has extended into the class-red region (and is dominated by the wrong class in this instance).*

Discriminant Adaptive Nearest-Neighbor (DANN)

- At each query point, a neighborhood of say 50 points is formed.
- Class probabilities are NOT assumed constant in this neighborhood. This neighborhood is used only to decide how to define the adapted metric.
- After the metric is decided, a normal k-nearest neighbor rule is applied to classify the query.
- The metric changes with the query.

- The DANN metric at a query point x_0 is defined by

$$D(x, x_0) = (x - x_0)^T \Sigma (x - x_0) \ ,$$

where

$$\begin{aligned} \Sigma &= \mathbf{W}^{-1/2}[(\mathbf{W}^{-1/2})^T \mathbf{B} \mathbf{W}^{-1/2} + \epsilon \mathbf{I}](\mathbf{W}^{-1/2})^T \\ &= \mathbf{W}^{-1/2}[\mathbf{B}^* + \epsilon \mathbf{I}](\mathbf{W}^{-1/2})^T \ . \end{aligned}$$

- $\mathbf{W}$ is the pooled within-class covariance matrix and $\mathbf{B}$ is the between class covariance matrix.

Intuition

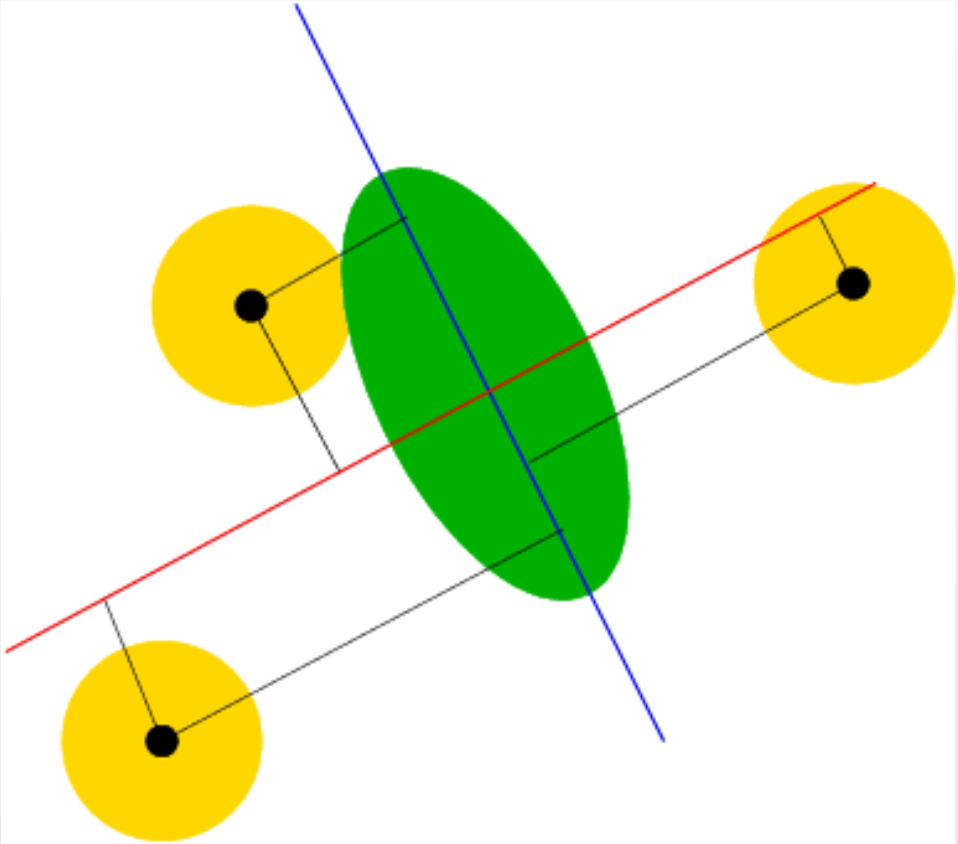
- We compute $\mathbf{W}$ and $\mathbf{B}$ in LDA.
- Recall we did similar computation when deriving discriminant coordinates.
- Eigen decomposition of $\mathbf{B}^* = \mathbf{V}^* D_B \mathbf{V}^{*T}$.

$$\begin{aligned}
\Sigma &= \mathbf{W}^{-1/2}[\mathbf{B}^* + \epsilon \mathbf{I}](\mathbf{W}^{-1/2})^T \\
&= \mathbf{W}^{-1/2}[\mathbf{V}^* D_B \mathbf{V}^{*T} + \epsilon \mathbf{I}](\mathbf{W}^{-1/2})^T \\
&= \mathbf{W}^{-1/2}\mathbf{V}^* \cdot (D_B + \epsilon \mathbf{I}) \cdot (\mathbf{W}^{-1/2}\mathbf{V}^*)^T
\end{aligned}$$

- Note that the column vectors of $\mathbf{W}^{-1/2}\mathbf{V}^*$ are simply the discriminant coordinates.
- $\epsilon \mathbf{I}$ is added to avoid using samples far away from the query point.

Geometric interpretation:

- To compute DNAA metric, $x - x_0$ is projected onto the discriminant coordinates.
- The projection values on the significant discriminant coordinates are magnified; those on the insignificant DCs are shrunk.
- The implication is that the neighborhood is stretched in the directions of the insignificant DCs and squeezed in those of the significant ones.
- $(\mathbf{W}^{-1/2})^T$ is introduced to sphere the data so that the within-class covariance matrix is the identity matrix.



- Significant discriminant coordinates represent directions in which class probabilities change substantially. Hence when we form a neighborhood, we want it to have small span in these directions. On the opposite, we want the neighborhood to have large span in directions in which class probabilities don't change much.
- To summarize: we want to form a neighborhood that contains as many samples as possible but in the mean while has approximately constant class probabilities.

- To summarize: we want to form a neighborhood that contains as many samples as possible but in the mean while has approximately constant class probabilities.

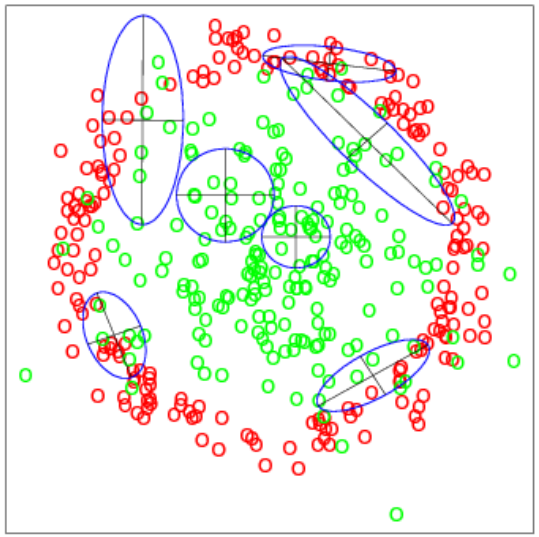


Figure 13.14: *Neighborhoods found by the DANN procedure, at various query points (centers of the crosses). There are two classes in the data, with one class surrounding the other. 50 nearest-neighbors were used to estimate the local metrics. Shown are the resulting metrics used to form 15-nearest-neighborhoods.*

Global Dimension Reduction

- At each training sample x_i, the between-centroids sum of squares matrix $\mathbf{B}_i$ is computed, using a neighborhood of say 50 points.
- Average $\mathbf{B}_i$ over all training samples:

$$\bar{\mathbf{B}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{B}_i \ .$$

- Let v_1, v_2, ..., v_p be the eigenvectors of the matrix $\bar{\mathbf{B}}$, ordered from largest to smallest eigenvalues θ_k. Then a rank-L, $L < p$, approximation to $\bar{\mathbf{B}}$ is

$$\bar{\mathbf{B}}_{[L]} = \sum_{l=1}^{L} \theta_l v_l v_l^T \ .$$

$\bar{\mathbf{B}}_{[L]}$ is optimal in the sense of solving the least square problem:

$$\min_{rank(M)=L} trace(\mathbf{B} - \mathbf{M})^2 \ .$$

And hence solves:

$$\min_{rank(M)=L} \sum_{i=1}^{N} trace(\mathbf{B}_i - \mathbf{M})^2 \ .$$