MSA220/MVE440 Statistical Learning for Big Data

Lecture 6

Rebecka Jörnsten

Mathematical Sciences
University of Gothenburg and Chalmers University of Technology

Predictive modeling

- Regression modeling: predictive modeling of continuous outcome variable y (or ordered, numerical)
- Classification: predictive modeling of a categorical outcome variable $y \in \{1, \cdots, C\}$.

Issues we care about

- Evaluate performance of the model
- Model selection, measures of variable importance

Evaluating performance

- Regression modeling: Usually prediction MSE (mean squared error)
- Classification: Often misclassification error rate
    - Make sure to check the error rate class by class!

Evaluating classification performance

In a two-class problem (1 vs 0), let's denote *tp* (true positive) the correctly predicted class 1 observations and *tn* the correctly predicted class 0 observations.
Likewise, let *fp* be the missed class 1 observations and *fn* the missed class 0 observations

- Accuracy: $\frac{tp+tn}{tp+tn+fp+fn}$
- Precision: $\frac{tp}{tp+fp}$ (how many of the observations you predict as 1 are truly 1?)

Evaluating classification performance

- Sensitivity: $\frac{tp}{tp+fn}$ (how many of the 1s did we predict as 1?) (TPR, true positive rate)
- Specificity: $\frac{tn}{tn+fp}$ (how many of the 0s were correctly predicted as 0?)
- False positive rate (FPR): 1-Specificity
- ROC curve: we can "play" with the tuning of our classifier, e.g. how easy it is to make a 1 decision and plot TPR vs FPR. A good classifier achieves high TPR for low FPR.
- AUC: often refers to the Area under the ROC curve as a summary measure of performance. An area of 1 is a perfect classifier, area 0.5 is a random classifier.
- AUC can also refer to $\frac{1}{2}(\frac{tp}{tp+fn} + \frac{tn}{tn+fp})$ is a measure of average error rate.

Evaluating classification performance

What about multi-class problems?

- Average Accuracy
- Average error rate
- Precision: $\frac{\sum_{c=1}^{C} tp_c}{\sum_{c=1}^{C} tp_c + fp_c}$
- Sensitivity: $\frac{\sum_{c=1}^{C} tp_c}{\sum_{c=1}^{C} tp_c + fn_c}$

How to evaluate performance

- Training, Validation and Testing
- Training and Validation: Data set used for training. Data set for Validation (tuning your method, e.g. model selection, method selection).
- Testing data set is left to the very end to estimate future performance of your method
- Training + Validation: advisable to perform MANY splits here to see how stable selection and error estimation is.

Classification

Lots of models - all of which in some form or another tries to estimate the local probability of an observation belonging to a certain class: $p(y = c|X = x) = *$.

- kNN: Finds a neighborhood $N(x)$ and estimates * by the neighborhood probabilities. Vote for the maximum probability class
- 0-1 regression: in a two-class problem, assume $p(y = c|X = x)$ is linear in x.
- logistic regression: assume $log(\frac{p(y=c|X=c)}{p(y\neq c|X=c)})$ is linear in x (multinomial model if more than 2 classes).
- CART: assume $p(y = c|X = x)$ is piecewise constant.
- Nearest centroid classifier: assume $p(y = c|X = x)$ only depends on the distance between x and the class means $\mu_c, c = 1, \cdots, C$.
- DA: assume $p(X = x|y = c)$ comes from a multivariate normal distribution $MVN(\mu_c, \Sigma_c)$ and use Bayes' rule $p(y = c|X = x) \propto p(X = x|y = c)\pi_c$ where π_c is the prior probability of class c
- ... and many many more.

Classification

Types of classifiers

- Probabilistic models: either logistic that focuses on $p(y|X)$ or discriminant analysis that focuses on $p(X, y)$ through a parametric framework.
- Rule-based: like CART, building decisions based on combinations of features. More general than CART would be to have overlapping rules with voting.
- Instance-based: forgo an explicit model for the data and instead base decisions by matching or comparing with other observations (like kNN). All about defining distance (weighted, adaptive, mixed data...)
- Support-vector machines: finding separating hyperplanes (possibly in a non-linear fashion) (more later) - loss functions...
- Neural-nets/Machine learning: rule-building without explicit definition of rule in terms of features.

Classification

Important ”special cases”

- Rare class or unbalanced data: upsampling, downsampling or weighting.
- Ensemble learning: bagging, model averaging, model stacking,...
- Semi-supervised learning: what if most of the data does not have labels? Ignore - or use? Self-training or combine unsupervised/supervised goals.

Classification - feature selection

Focus on upcoming lecture/Mini

- Filter - select features based on some property
- Wrapper - build into classification algorithm
- Embedding - make selection part of the goal of the method

Classification - feature selection

Filter methods

- F-test or similar (Between/Within class variation) - what if many classes? imbalance?
- information theoretic: reduction in entropy (spread) when we condition on class label - generalization to mixed, discrete data - but see above
- Based on distances to observations with same label and average across other labels (ReliefF) - local query.
- Many variants.... careful how much they impact specific classifiers - they are univariate selectors and make assumption about relevance.

Classification - feature selection

Wrapper methods

- Incorporate selection into the classification method - e.g. by evaluating performance as function of selection.
- How search for set to evaluate? All-subset = expensive, Forward/backward = greedy, hybrid/random = needs many iterations

Classification - feature selection

Embedding

- Introduce a *penalty* on the number of features used - that's a hard selection problem...
- ... which we make practical by using sparsity constraints on feature-specific coefficients: L1-penalty (lasso, and variants thereof)!
- We will look at this first in the context of discriminant analysis.

Recap: Nearest centroid classifier

We compute

$$\hat{\mu}_c = \frac{1}{N_c} \sum_{y_i = c} x_i$$

where N_c is the number of observations in class c. That is, we compute the mean, or centroid, of each class.
The rule is

$$\hat{c}(x) = \arg\min_c d(x, \hat{\mu}_c)$$

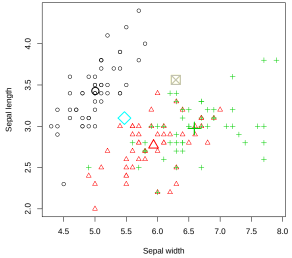
where $d(.,.)$ is the distance between observation location x and the centroid μ. This is usually the euclidean distance

$$d(x, \hat{\mu}_c) = || x - \hat{\mu}_c ||^2 = (x - \hat{\mu}_c)'(x - \hat{\mu}_c).$$

The rule is thus to allocate each observation to the class with the closest centroid.

Nearest centroid classifier

Problems?



From the above examples it is clear that one needs to consider both *spread/scale* of a distribution (the amount of spread around a centroid) and the *shape* of the distribution (the correlation structure between the features) to form a good classification rule.

Discriminant analysis

General setup is the following;

- prior $\pi_c = p(y = c)$
- data distribution $p(x \mid y = c) \sim N(\mu_c, \Sigma_c)$ where μ_c is a p-dimensional vector and Σ_c is a p-by-p dimensional covariance matrix.

Discriminant analysis

The multivariate normal assumption leads to the following simple, intuitive parameter estimates:

- $\hat{\pi}_c = N_c/N$, where $N_c = \sum_i 1\{y_i = c\}$ is the number of observations in class c.
- $\hat{\mu}_c = \frac{1}{N_c} \sum_{y_i=c} x_i$
- $\hat{\Sigma}_c = \sum_{y_i=c} (x_i - \hat{\mu}_c)(x_i - \hat{\mu}_c)'/(N_c - 1)$

This is quite a large number of parameters...: $(C-1)$ for $\hat{\pi}$ (not C since the πs add to 1), $p \times C$ mean parameters, and $p(p+1) \times C/2$ covariance parameters (since they're symmetric). As the dimensionality of the problem grows (p) the number of parameters grows quickly, especially due to the covariance matrices.

Linear discriminant analysis

The solution to this problem is to try to simplify the modeling assumption somewhat: $\Sigma_c = \Sigma$, same correlation structure between the features for all classes.

- Realistic? Think about the heart disease data. Do you think ldl-level and bmi are correlated the same way for healthy patients and patients with heart disease?
- The assumption may not be realistic BUT in statistics you always have to worry about the flexible methods suffering from poor estimation and thus leading to a bad classifier. Here, the approximation of equal correlation may be "safer" than trying to build a very complex method with many parameters on noisy data or a data with a small sample size.
- Under this assumption you get $\hat{\Sigma}$ from a *pooled* estimate.

$$\hat{\Sigma} = \sum_{c=1}^{C} \sum_{y_i=c} (x_i - \hat{\mu}_c)(x_i - \hat{\mu}_c)'/(N - C) = \sum_{c=1}^{C} \hat{\Sigma}_c \frac{N_c - 1}{N - C},$$

a weighted average of covariance estimates from each individual class.

Other variants of discriminant analysis

- $\Sigma_c = \Lambda_c$, diagonal matrix. You ignore the correlations between features. (DQDA) (Naive Bayes)
- $\Sigma_c = \Sigma = \Lambda$, diagonal matrix. You ignore correlations AND make the feature variance the same for all classes. (DLDA)
- $\Sigma_c = \Sigma = \sigma^2 I$, nearest centroid. Here you ignore all differences between classes and features in terms of variance and ignore feature correlations.
- Σ_c: QDA
- $\tilde{\Sigma}_c$ a regularized estimate between Σ_c, Σ and $diag(\Sigma_c)$: RDA

Building the classifier

We define the boundary between two classes, l and c, at the x-locations where the posterior probabilities are equal:

$$\{x : \pi_l p(x \mid y = l) = \pi_c p(x \mid y = c)\}$$

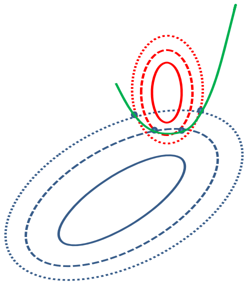
Equivalently, we can write this on a log-scale as

$$\{x : \log \frac{p(x \mid y = c)}{p(x \mid y = l)} + \log \frac{\pi_c}{\pi_l} = 0\}$$

If we plug the MVN models into this we get an expression *linear in* x for LDA and *quadratic in* x for QDA.

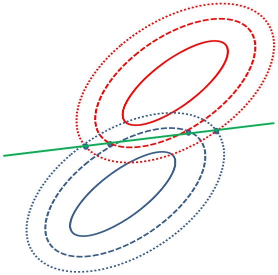
Building the classifier

To draw these boundaries, simply look for points in x-space where the posterior distributions have the same value in two classes. I have illustrated that below with two classes and different line types corresponding to the contours for 90, 95 and 99 percent of the probability mass.



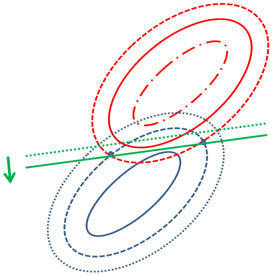
Building a linear classifier

To draw these boundaries, simply look for points in x-space where the posterior distributions have the same value in two classes. I have illustrated that below with two classes and different line types corresponding to the contours for 90, 95 and 99 percent of the probability mass.



Building a linear classifier

So what is the role of the prior? The prior will simply shift the contours of the data distribution center-outward if it increases, resulting in intersections with other class distribution contours further away from the distribution with a higher prior.



Mixture DA

A very nice alternative to QDA that generalizes LDA to more flexible boundaries is *mixture discriminant analysis* (MDA), introduced by Hastie and Tibshirani in the mid-90s.
We make the classifier more complex by allowing each class to be made up of many, simple components (as opposed to one complex component as in QDA). By combining many simple shapes we can build up quite complex shapes in x-space! Example: you can build a donut shape in x-space with 5-6 spherical distributions.

Mixture DA

We assume the following model for each class

$$p(x \mid y = c) = \sum_{r=1}^{R_c} \pi_{cr} N(x; \mu_{cr}, \Sigma)$$

Notice

- There are R_c components for class c and this may differ from class to class
- Each component has a different contribution or "weight" in the class distribution, π_{cr}
- Each component within and between the classes have the same shape, Σ.

For large p, the last bullet constitutes a large savings in terms of the number of parameters compared to QDA.

Validation

Both RDA and MDA can be tuned to be more or less flexible/local.
As always in statistics - we have to consider the bias-variance trade-off!
Use cross-validation to select.

More on Discriminant analysis

As mentioned in the previous lectures, one problem with LDA stems from the instability of the estimate $\hat{\Sigma}$ when n is small and/or p is large and/or x-features are correlated.

Flexible DA methods

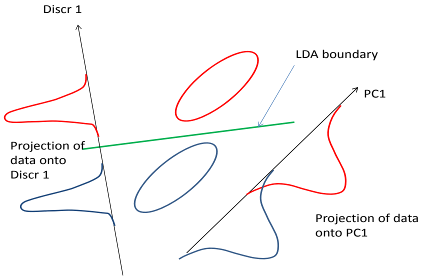
Penalized and regularized DA addresses one problem with LDA and QDA, poor performance due to unstable estimates of $\hat{\Sigma}^{-1}$ (high variance) by shrinking the Σ_c estimates to common Σ or to a diagonal matrix.
We also need to be concerned with potential BIAS, meaning the linear or quadratic boundaries that LDA and QDA implicitly assume are too simplistic to separate the classes from eachother.

Reduced rank methods

PDA tries to fix the problem with LDA's high variance by penalizing the estimate $\hat{\Sigma}$ as $\hat{\Sigma} + \lambda I$.
Another way to deal with this problem is to use a reduced rank approximation of Σ instead. That is, use the leading principal components only!
Is this a good idea? Often, yes - BUT you have to be very careful.
It is quite possible that the class means are not well separated along the leading PC directions of $\hat{\Sigma}$.

Reduced rank methods

Look at the distribution of the data when projected onto the first principal component. All the variation is carried with the data onto the projection and the distributions overlap, meaning classification is not going to be perfect.



Reduced rank methods

What does LDA do then? We utilize the distance

$$(x - \hat{\mu}_c)'\hat{\Sigma}^{-1}(x - \hat{\mu}_c) = (U'(x - \hat{\mu}_c))'D^{-1}(U'(x - \hat{\mu}_c))$$

where $\hat{\Sigma} = UDU'$.
As discussed before, this means we *sphere* the data and use the nearest centroid in the new coordinate system with data $\tilde{x} = D^{-1/2}U'x$.
If we use only the leading principal components that is equivalent to taking u_1, u_2 from $U = (u_1, u_2, \cdots, u_p)$ and creating a new, lower-dimensional data set comprising $\tilde{x}_1 = d_1^{-1}xu_1$ and $\tilde{x}_2 = d_2^{-1}xu_2$.

Reduced rank methods

R.A. Fisher had the following idea: What if you project onto a lower dimensional space and do the classification there?
The goal: *Find the optimal projection in* $\leq C - 1$*-space such that the centroids are as spread out as possible while the within-class variance (variance around centroids) is as small as possible*.
The primary goal is the centroid spread and the secondary goal you are also trying to fulfill is the limitation of the within-class spread.
Note - this is NOT equivalent to PCA! (see figure on earlier slide).

Fisher's LDA - FLDA

Some notation:

- $\Sigma = \Sigma_W$ is the within-class variance.
- Σ_B is the *between-class* variance, the spread of the centroids

We define these as

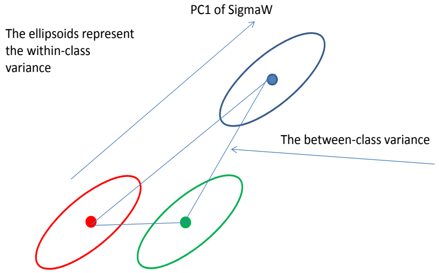
$$\Sigma_W = \sum_{c=1}^{C} \sum_{y_i = c} (x_i - \hat{\mu}_c)(x_i - \hat{\mu}_c)' / (N - C)$$

and

$$\Sigma_B = \sum_{c=1}^{C} (\hat{\mu}_c - \bar{x})(\hat{\mu}_c - \bar{x})' / (C - 1)$$

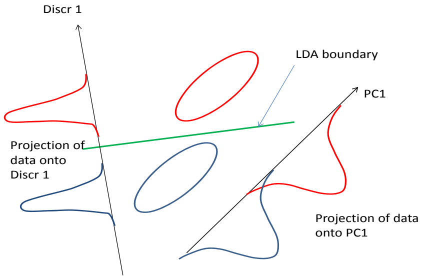
Fisher's LDA - FLDA

Here's an illustration of the between-class variance Σ_B and the within-class variance Σ_W


PC1 of SigmaW
The ellipsoids represent the within-class variance
The between-class variance

FLDA

FLDA means finding a projection other than the leading PCs such that the centroids are spread when taking the within-class variance into account.



FLDA

Mathematically, Fisher's problem can be written as
Find directions *a* such that

$$\max_{a} \frac{a'\Sigma_B a}{a'\Sigma_W a}$$

Fisher's problem simply states we want to maximize the centroids spread compared to the within-class spread.

FLDA

The ratio

$$\frac{a'\Sigma_B a}{a'\Sigma_W a}$$

is called the Rayleigh quotient.
How do we go about maximizing this for not one, but several directions a (since 1-D projections may not suffice to separate $C > 2$ classes.

FLDA

We write the problem as

$$\max_a a'\Sigma_B a \ \ \textit{subject to} \quad a'\Sigma_W a = I$$

Note, our primary goal is maximizing the between-class spread.
Our secondary goal is represented as a contraint where we say the directions should sphere the data.
We can rewrite this using Lagrangian methods as

$$\min_a -\frac{1}{2}a'\Sigma_B a + \frac{1}{2}\lambda(a'\Sigma_W a - I) = ***$$

where λ is the Lagrangian parameter.

FLDA

We find the minimizer by taking derivatives and setting to 0:

$$\frac{\partial ***}{\partial a} = -\Sigma_B a + \lambda \Sigma_W a = 0$$

We can write

$$\Sigma_B a = \lambda \Sigma_W a$$

or

$$(\Sigma_W^{-1} \Sigma_B) a = \lambda a$$

FLDA

$$(\Sigma_W^{-1}\Sigma_B)a = \lambda a$$

looks just like an eigenvalue problem!
That means that the optimal directions for separating the class centroids are the vectors a that are eigenvectors of the matrix $\Sigma_W^{-1}\Sigma_B$.
There's one problem - this matrix is not symmetric. Therefore this is not a standard eigenvalue problem but requires a *generalized eigendecomposition*.

FLDA

Solution to

$$(\Sigma_W^{-1}\Sigma_B)a = \lambda a$$

Write $\Sigma_W = \Sigma_W^{1/2}\Sigma_W^{1/2}$ (which you do by defining $\Sigma = UDU'$ and $D = D^{1/2}D^{1/2}$ as before).
Plug in above to obtain

$$(\Sigma_W^{-1/2}\Sigma_B\Sigma_W^{-1/2})(\Sigma_W^{1/2}a) = \lambda(\Sigma_W^{1/2}a) =$$

$$(\Sigma_W^{-1/2}\Sigma_B\Sigma_W^{-1/2})w = \lambda w$$

which is a standard eigenproblem for w since the matrix $(\Sigma_W^{-1/2}\Sigma_B\Sigma_W^{-1/2})$ is symmetric.

FLDA

You solve for the eigenvectors w and compute the original vectors that solve Fisher's problem as

$$v = \Sigma_W^{-1/2} w$$

These are the directions to project onto to separate the class means as far as possible given the within-class variance!

FLDA

- FLDA finds the optimal subspace separation of the centroids given within-class variance
- It corresponds to an eigendecomposition of $\Sigma_W^{-1}\Sigma_B$
- The data projected onto these eigenvectors are called *discriminant variables*
- Reduced dimension or reduced rank LDA means that you use only the leading eigenvectors of $\Sigma_W^{-1}\Sigma_B$

Flexible discriminant analysis, FDA

In the mid-90's the Stanford group (Trevor Hastie, Robert Tibshirani and Andreas Buja and others) used Mardia's *Optimal Scoring* to reformulate DA as a regression problem.
This had two huge benefits:

- Flexible extensions: use polynomial and other more complex regression models to augment LDA
- Regularization: use penalized and sparse regression methods (feature selection) to reduce the variance of LDA

Flexible discriminant analysis, FDA

First, let's review how the FLDA discriminant variables are computed.

- Compute the centroid matrix $M = (\hat{\mu}_1, \hat{\mu}_2, \cdots, \hat{\mu}_C)$ which is a $C \times p$ matrix.
- "Sphere" the μs with the within-class covariance, Σ_W: $M^* = M\Sigma_W^{-1/2}$
- Compute the between-variance of the sphered centroids:

$$\Sigma_B^* = Cov(M^*) = (M^* - \bar{x}^*)(M^* - \bar{x}^*)'/C - 1$$

- The eigendecomposition of $\Sigma_B^* = V^* D_B V^{*'}$ provide the optimal discriminant vectors (eigenvectors of $\Sigma_W^{-1}\Sigma_B$)

Optimal Scoring

1. Create a $N \times C$ matrix Y where column k has 1s for observations i belonging to class k and 0 elsewhere.
2. Regress Y on the data matrix X $(N \times p)$ using least squares.
   - Results in a least-squares coefficient matrix B $(p \times C)$ where $\hat{B} = (X'X)^{-1}X'Y$
   - Quick recap of regression: want to find B to minimize $|| Y - XB ||^2 = (Y - XB)'(Y - XB)$
     Take derivatives with respect to B and set to 0:
     $-2X'(Y - XB) = 0$ and solve for B
   - The fitted values (values on the regression lines) are given by $\hat{Y} = X\hat{B} = X(X'X)^{-1}X'Y = HY$ where the hat-matrix $H = X(X'X)^{-1}X'$ $(N \times N)$ is a project of Y onto the space spanned by X.

Optimal Scoring

3 Perform an eigendecomposition of $Y'\hat{Y} = Y'HY$

$$(Y'HY)\theta = \lambda\theta$$

One can show that θ is directly proportional to the discriminant vectors V we defined before!

Optimal Scoring

To see this, compute $X'X$ and $X'Y$.
$X'X = \Sigma_W$ except for a normalizing factor.
$X'Y = M$, the $p \times C$ matrix of class centroids (except for a normalizing factor).
(Try this yourself by writing out the $0/1$ Y-matrix and the data matrix X).
Now, $\hat{B} = (X'X)^{-1}X'Y = \Sigma_W^{-1}M$ and therefore it follows that

$$Y'\hat{Y} = Y'X(X'X)^{-1}X'Y = M'\Sigma_W^{-1}M =$$

$$= (\Sigma_W^{-1/2}M)'(\Sigma_W^{-1/2}M) = M^{*'}M^* = \Sigma_B^*$$

That is, the eigendecomposition problem in optimal scoring is the eigendecomposition of the between-centroid spread in the new coordinate system (taking within-class variance into account) = Same thing as FLDA!!!

Optimal Scoring

Why bother?
The point is that once we turn this into a regression problem it is easy to see how we can come up with extensions t the method.
You can use polynomial regression, nonlinear regression, semi-parametric regression, regressions with priors.... anything you want!
This idea is called *Flexible Discriminant Analysis*

Regularized Discriminant Analysis

We already talked about one particular kind of penalized DA: when we used the inverse of $\hat{\Sigma}_W + \lambda I$ to rotate/sphere our data.
Our regression analogue above connects this approach to penalized (or ridge) regression. One can run other kinds of penalized regression schemes also. For example, you could do feature selection at the regression step via e.g. lasso.
This method, and variants on the same theme, is called *sparse discriminant analysis*.

Sparse Discriminant Analysis

In sparse LDA we apply an $L1$ penalty to the regression coefficients in the optimal scoring problem. The final rule will now consist of discriminant vectors based on a subset of variables.
As in standard FLDA we can plot the data and classify the data in a reduced dimensional space based on these sparse discriminant vectors.
`sparseLDA` package

Regularized Discriminant Analysis

Several methods have been proposed for regularizing the within-covariance estimates.

- In QDA we can penalize each individual within-class covariance toward a common covariance (LDA)
- We can regularize the common within-class covariance toward a diagonal matrix (RDA)
- We can assume that the within-covariance matrix *is* diagonal (naive bayes)
- We can use a ridge-penalized estimate of the covariance matrix (PDA)

High-dimensional classification

Going back to the original "fix" - trying the regularize the within-class covariance estimates. There have been other types of proposals here as well that also uses sparse modeling.

- Estimate the inverse covariance using sparse modeling (`graphical lasso`)
- Use sparse inverse covariance estimation and approximate this with a block-diagonal matrix (Pavlenko et al, 2008). This is like a less severe approximation than naive bayes.
- Use your regularized estimate of the within-covariance in your DA rule.

NB and Shrunken centroids

A special case of Naive Bayes is to replace the within-covariance estimate by its diagonal component.
This means we assume that features are independent.
In high-dimensional settings this tends to work quite well! The classifier now works on each variable at a time

$$k(i) = \arg\min_{l} \sum_{j=1}^{p} \frac{(x_{ij} - \mu_{lj})^2}{\sigma_l^2}$$

where $k(i)$ is the optimal class for observation i.
Tibshirani et al (2002) proposed we not use all the variables for classification.

- Shrink the class means (centroids) toward a common value (after standardizing by the within-class standard deviation)
- We can regularize the common within-class covariance toward a diagonal matrix (RDA)
- We can assume that the within-covariance matrix *is* diagonal (naive bayes)

Shrunken centroids

- Use a diagonal covariance estimate $diag(\Sigma + s_0^2 I)$ (where a small s_0 is used to avoid having really small standard deviations in the denominator later on)
- Compute for each variable j

$$t^*_{kj} = \frac{\hat{\mu}_{kj} - \hat{\mu}_j}{m_k(s_j + s_0)}$$

where $\hat{\mu}_j$ is the overall mean for variable j, $s_j = \hat{\Sigma}_{jj}$ and $m_k = \sqrt{\frac{1}{n_k} + \frac{1}{n}}$

- Apply a soft-threshold to t^*_{kj}: $t_{kj} = ST(t^*_{kj}, \Delta)$
- Define $\mu^s_{kj} = \mu_j + m_k(s_j + s_0)t_{kj}$
- Use these *shrunken centroids* in your classifier!
- `pamr` package

SC-RDA

In 2005, Guo et al, proposed a combination of RDA and shrunken centroids

- We can use the SC method to shrink the centroids, either in the original data space
- or in the rotated data space!
- $\arg\min_k (x - \mu_k^s)' \hat{\Sigma}_R^{-1} (x - \mu_k^s)$
- `rda` package