Lecture 8: Boosting, SVMS and flexible discriminant analysis

Rebecka Jörnsten, Mathematical Sciences

MSA220/MVE441 Statistical Learning for Big Data

4th April 2025

Tuning boosting machines

- At the end of last lecture we had a framework for creating boosting models
- The key is how we train them in practice
- To avoid overfitting you need to utilize early stopping, using a validation data to track the performance of the generated models
- Other tuning mechanisms include the form of the weak learners, usually trees that are shallow.
- You can also choose the learning rate to control how aggressively you update your models in each iteration
- Optimizing the step length requires a line search, and updating less than the full step is a form of regularization.

XGBoost

- GBM can be slow to train if your data is big and/or high-dimensional.
- XGBoost was introduced by T. Chen and C. Guestrin in 2016 as a fast implementation but also with a different take on the actual boosting step itself (Taylor approximation).

Let's state the boosting problem:

$$\min_{\alpha, f_t} L(y, F_{t-1} + \alpha f_t)$$

We approximate the loss function as follows, thinking of the update as a small perturbation of the previous step

$$L(y, F_{t-1} + \alpha f_t) \simeq L(y, F_{t-1}) + \nabla_{F_{t-1}}(L)\alpha f_t$$

This linear approximation holds if the update is small.

$$L(y, F_{t-1} + \alpha f_t) \simeq L(y, F_{t-1}) + \nabla_{F_{t-1}}(L)\alpha f_t$$

This linear approximation is what we used in GBM. Think of the second term as the inner product of the functional gradients with your weak learner - in GBM we maximize this correlation by training a weak learner to approximate the gradients.

In XGBoost, we use a 2nd order expansion to approximate the loss:

$$L(y, F_{t-1} + \alpha f_t) \simeq L(y, F_{t-1}) + \nabla_{F_{t-1}}(L)\alpha f_t + \frac{1}{2}(\nabla^2_{F^2_{t-1}} L)(\alpha f_t)^2$$

This translates to searching for the optimal update via Newton-Rhapson instead of gradient descent.

XGBoost

- The XGBoost contains other bells and whistles, primarily a very fast implementation for creating the weak learners (building the trees) which is useful for the high-dimensional cases, building deeper trees and pruning them - so the weak learners can take on different complexities adaptively.
- In XGBoost and other GBM implementations, you can use *drop-out* - choosing to ”forget” the early weak learners as another form of regularization mechanism (in case early steps leads you in the direction of local optima).
- Applying the methods to stochastically generated subsamples also helps with convergence, and can prevent overfitting.

Support Vector Machines

Separating hyperplanes

- Let $y \in \{-1, 1\}$ and define your classifier as the linear model $f(x) = \beta_0 + x'\beta$ with prediction $sign(f(x))$
- That is, the decision boundary is defined as $\{x : x'\beta + \beta_0 = 0\}$ and you classify $\hat{y} = 1$ if you are on the positive side of the decision boundary $f(x) > 0$ and -1 otherwise
- That is, $y_i f(x_i) > 0$ for correct predictions and < 0 otherwise (cf AdaBoost from last lecture)

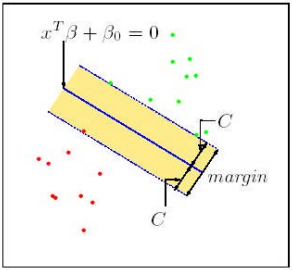


figure from book

Separating hyperplanes

- If the classes are well separated there are infinitely many decision boundaries that separate them: you can rotate the boundary and shift it as long as no observation transitions the boundary.
- To obtain a unique model and one that is also likely to generalize well to future data, we consider the boundary with the largest *margin* to the data (C in the figure).

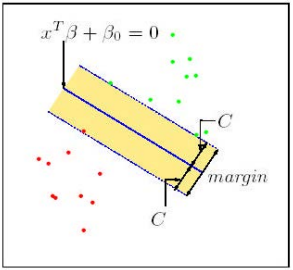


figure from book

Separating hyperplanes

- We want to find the decision boundary that maximizes this margin, i.e. the observation distances to the boundary.
- The boundary is defined as $\{x : x'\beta + \beta_0 = 0\}$. Note, that the vector β is by definition orthogonal to this boundary.
- Thus is we want to compute the distance from any observation to the boundary, we should project the distance vector onto the vector $\beta/||\beta||$. For any point x_0 on the boundary we have $x_0'\beta = -\beta_0$ and thus the distance to the boundary for any observation x is

$$< x - x_0, \frac{\beta}{||\beta||} >= \frac{x'\beta - x_0'\beta}{||\beta||} = \frac{x'\beta + \beta_0}{||\beta||} = \frac{f(x)}{||\beta||}$$

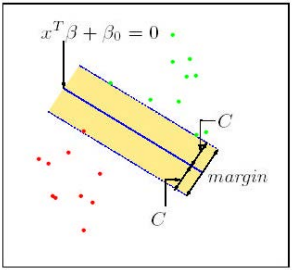


figure from book

Separating hyperplanes

- Let us now formulate the problem where we want to maximize the distances to the boundary: we want all observations to be on the right side of the boundary and at least distance C from it.

$$\max_{\beta,\beta_0} C$$

$$s.t. \frac{1}{||\beta||} y_i(x_i'\beta + \beta_0) \geq C, \forall i$$

- The product with y_i ensures we check that we are on the right side of the boundary for the distances we compute.
- Equivalently, write the constraints as $y_i(x_i'\beta + \beta_0) \geq C||\beta||, \forall i$

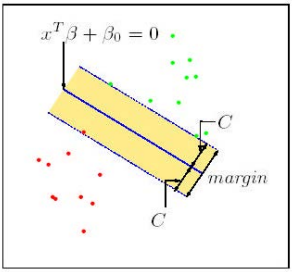


figure from book

Separating hyperplanes

- Since the norm of β plays an equal role on both sides of the $\geq$, the scale of the β doesn't really matter - just compute the constraint with $\gamma = c\beta$ to see this.
- Let us simply choose β such that $||\beta|| = 1/C$, then the optimization problem above is equivalent to

$$\min_{\beta,\beta_0} \frac{1}{2}||\beta||^2$$

$$s.t. y_i(x_i'\beta + \beta_0) \geq 1$$

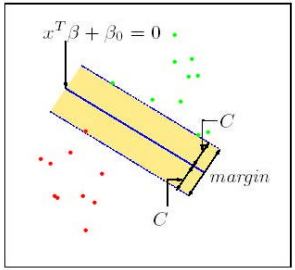


figure from book

Separating hyperplanes

- Lagrangian:

$$\frac{1}{2}||\beta||^2 - \sum_i \alpha_i \big(y_i(x_i'\beta + \beta_0) - 1\big)$$

- Take derivatives and set to 0:

$$\frac{dL}{d\beta} = \beta - \sum_i \alpha_i y_i x_i$$

$$\frac{dL}{d\beta_0} = \sum_i \alpha_i y_i$$

- Let's plug this into the equation above.

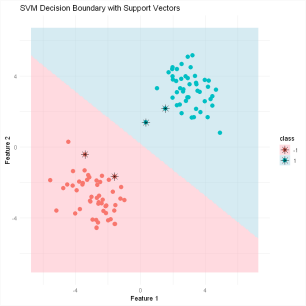


figure from demo

Separating hyperplanes

- Lagrangian:

$$\frac{1}{2}||\beta||^2 - \sum_i \alpha_i\big(y_i(x_i'\beta + \beta_0) - 1\big) =$$

$$= \frac{1}{2}\sum_{i,k}\alpha_i y_i x_i' x_k y_k \alpha_k - \sum_i \alpha_i y_i x_i'\Big(\sum_k \alpha_k y_k x_k - \underbrace{\sum_k \alpha_k y_k \beta_0}_{=0}\Big) + \sum_i \alpha_i$$

$$= \sum_i \alpha_i - \frac{1}{2}\sum_i\sum_k \alpha_i\alpha_k y_i y_k (x_i' x_k)$$

$$s.t. \alpha_i \geq 0, \sum_i \alpha_i y_i = 0$$

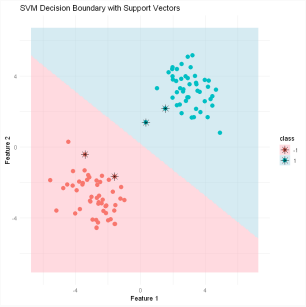


figure from demo

Separating hyperplanes

- Primal:

$$\frac{1}{2}||\beta||^2 - \sum_i \alpha_i\big(y_i(x_i'\beta + \beta_0) - 1\big) =$$

- Dual:

$$= \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_k \alpha_i \alpha_k y_i y_k (x_i' x_k)$$

$$s.t. \alpha_i \geq 0, \sum_i \alpha_i y_i = 0$$

- Maximize wrt α_i s.t. these constraints and the KKT conditions: constraints above and $\alpha_i\big(\big(y_i(x_i'\beta + \beta_0) - 1\big) = 0$ (compl.slackness). If the constraint is violated this is not a feasible solution.

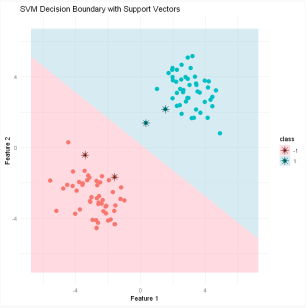


figure from demo

Separating hyperplanes

- Primal:

$$\frac{1}{2}||\beta||^2 - \sum_i \alpha_i \big(y_i(x_i'\beta + \beta_0) - 1\big) =$$

- Dual:

$$= \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_k \alpha_i \alpha_k y_i y_k (x_i' x_k)$$

$$s.t. \alpha_i \geq 0, \sum_i \alpha_i y_i = 0$$

- Notice, if $\alpha_i > 0$ then to satisfy the constraints we must have $y_i(x_i'\beta + \beta_0) = 1$ - that is, the observations that are on the margin!
- Likewise, if $y_i(x_i'\beta + \beta_0) > 1$, then we need $\alpha_i = 0$

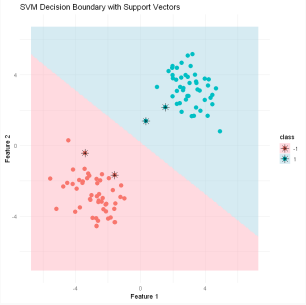


figure from demo

Separating hyperplanes

- Primal:

$$\frac{1}{2}||\beta||^2 - \sum_i \alpha_i \big(y_i(x_i'\beta + \beta_0) - 1\big) =$$

- Notice, if $\alpha_i > 0$ then to satisfy the constraints we must have $y_i(x_i'\beta + \beta_0) = 1$ - that is, the observations that are on the margin!
- Likewise, if $y_i(x_i'\beta + \beta_0) > 1$, then we need $\alpha_i = 0$
- The observations for which $\alpha_i > 0$ are called the *support vectors* and are the only observations you need to determine the boundary.

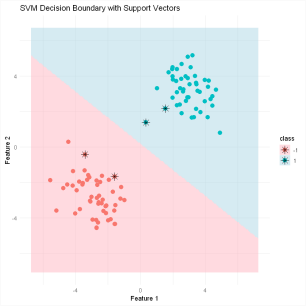


figure from demo

Support Vector Machines

- What if the classes overlap? Then, we need to ”soften” the constraints and define the optimal boundary to allow for observations to encroach onto the margin or even transition the boundary.
- To handle this we introduce *slack variables*, ξ_i with corresponding constraint

$$y_i(x_i'\beta + \beta_0) \geq C(1 - \xi_i)$$

- We add a constraint on the sum $\sum \xi_i$, which measures how much we ”soften” up the problem.

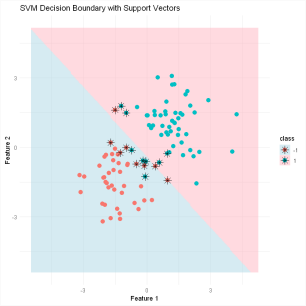


figure from demo

Support Vector Machines

- Following the same steps as in the separated case, setting $||\beta|| = 1/C$ we formulate the problem as

$$\frac{1}{2}||\beta||^2$$

$$s.t. y_i(x_i'\beta + \beta_0) \geq 1 - \xi_i$$

$$\xi_i \geq 0, \sum_i \xi_i \leq constant$$

where the constant controls have much slack you introduce.

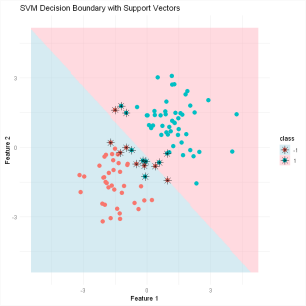


figure from demo

Support Vector Machines

- As before, we formulate the lagrangian setup:

$$\frac{1}{2}||\beta||^2 + K\sum_i \xi_i - \sum_i \alpha_i\big(y_i(x_i'\beta + \beta_0) - (1-\xi_i)\big) - \sum_i \mu_i\xi_i$$

- Taking derivatives wrt β, β_0, ξ_i and settings to zero we get

$$\beta = \sum_i \alpha_i y_i x_i, 0 = \sum_i \alpha_i y_i, K - \alpha_i - \mu_i = 0,$$

$$\alpha_i = K - \mu_i, \alpha_i \geq 0, \mu_i \geq 0, \xi_i \geq 0$$

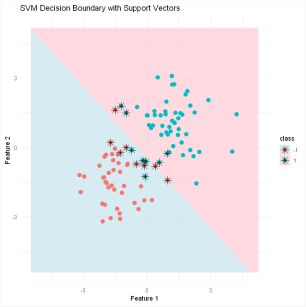


figure from demo

Support Vector Machines

- Plug in and formulate the dual problem

$$\sum_i \alpha_i - \frac{1}{2} \sum_i \sum_k \alpha_i \alpha_k y_i y_k (x_i' x_k)$$

and maximize wrt $\alpha_i \in [0, K]$, s.t. $\sum_i \alpha_i y_i = 0$ in addition to the KKT conditions (checking feasibility of solution)

$$\alpha_i \big(y_i (x_i' \beta + \beta_0) - (1 - \xi_i) \big) = 0$$

$$\mu_i \xi_i = 0$$

$$y_i (x_i' \beta + \beta_0) - (1 - \xi_i) \geq 0$$

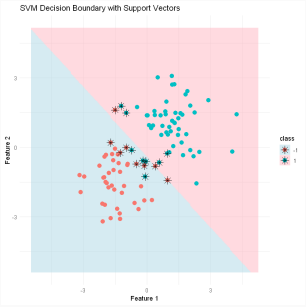


figure from demo

Support Vector Machines

- Similarly to before, the KKT condition

$$\alpha_i\big(y_i(x_i'\beta + \beta_0) - (1 - \xi_i)\big) = 0$$

leads to a sparse solution in the observation set where only the observations on the margin and those with slack variables $\xi_i > 0$ contribute to the solutions for the boundary defined by $\beta = \sum \alpha_i y_i x_i$

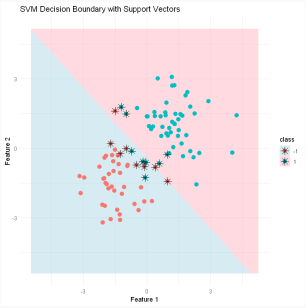


figure from demo

SVMs and the kernel trick

- If we expand the feature space to $\phi(x)$, notice how in the dual formulation, only the inner product appears:

$$\sum_i \alpha_i - \frac{1}{2} \sum_i \sum_k \alpha_i \alpha_k y_i y_k \phi(x_i)' \phi(x_k)$$

and with

$$\beta = \sum_i \alpha_i y_i \phi(x_i)$$

$$f(x) = \phi(x)'\beta + \beta_0 = \sum_i \alpha_i y_i \phi(x)' \phi(x_i) = \sum_i \alpha_i y_i k(x, x_i)$$

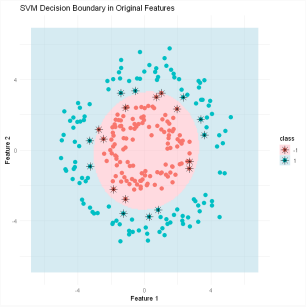


figure from demo

SVMs and the kernel trick

- That means we can make SVMs very flexible through the kernel trick!
- One can also formulate the SVM problem through a different loss function with regularization on the β:

$$\sum_i [1 - y_i f(x_i)]_+ + \frac{1}{\lambda}||\beta||^2$$

- This is called the hinge loss.
- A more ”statistical” type derivation by going this route.

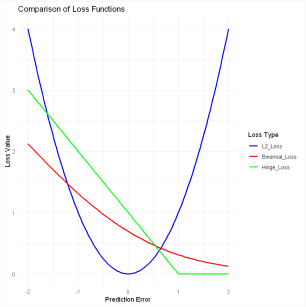


figure from demo

Discriminant analysis revisted

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:

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

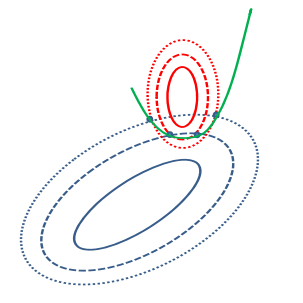
Equivalently, we can write this on a log-scale as

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

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

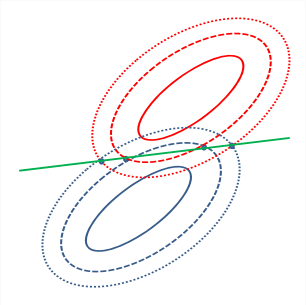
Building the classifier

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.



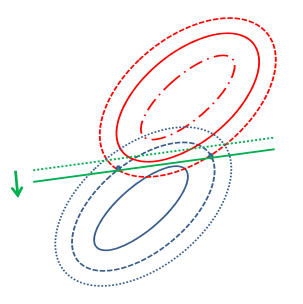
Building a linear classifier

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

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

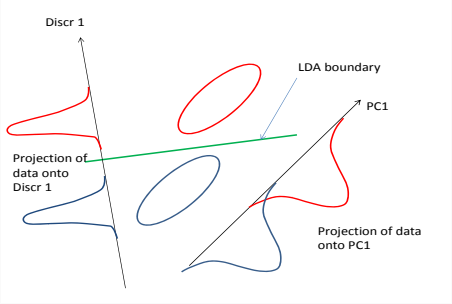
Notice

- 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.

Fisher's discriminant analysis - 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.



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

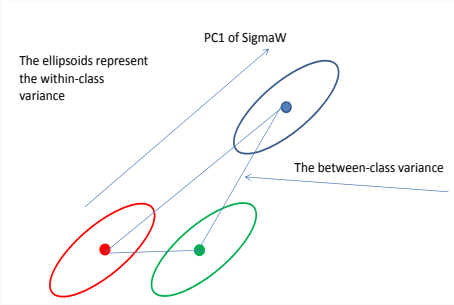
$$\Sigma_W = \sum_{c=1}^{C} \sum_{y_i=c} (x_i - \hat{\mu}_c)(x_i - \hat{\mu}_c)'/(N - C)$$

and

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

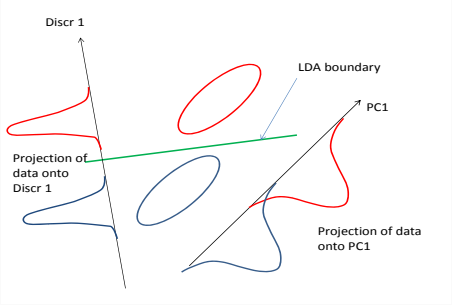
Fisher's LDA - FLDA

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



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.



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.

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.

We write the problem as

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

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.

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

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

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

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*.

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.

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.

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 to 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* - packages available!

kernel DA

We can make DA flexible in another way as well.
What if we go back to the problem formulation:

$$\max_a a'\Sigma_B a \;\; subject\ to \;\; a'\Sigma_W a = I$$

Let's expand the feature space to $\phi(x)$ and compute the between Σ_B^ϕ and within Σ_W^ϕ in this space instead.
Luckily, this can be formulated in terms of the kernel again! (see the paper on canvas). Really neat: pseudo-means in terms of kernel densities provide the between-covariance, and within-class centered kernel distances provide the within-covariances!