# MSA220/MVE440 Statistical Learning for Big Data

## Lecture 9

**Rebecka Jörnsten**

**Mathematical Sciences**
**University of Gothenburg and Chalmers University of Technology**

# RECAP!!!

We have covered A LOT of material in the past few weeks so I thought it best we take a pause and revisist methods and issues we've talked about.

- 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.

- We have focused on Least Squares as the fitting mechanism
- Simple and easy to use BUT don't forget to check 5 basic assumptions!
- $\hat{\beta} = (X'X)^{-1}X'y$, $\hat{y} = X(X'X)^{-1}X'y = Hy$
- Closed-form solution. Fitted value is a projection of $y$ onto the linear space spanned by $X$
- Can generalize to other type of projection models where $H = H(X)$, e.g. smoothers, polynomial models,...
- LS: a lot depends on how well the inverse $(X'X)^{-1}$ behaves

- $p > n$ we can't even take the inverse of $X'X$
- and not if $x$s are correlated.

What to do?

- Forward selection
- Reduce dimensionality of $X$ by some pre-screening
- Use regularized regression
- ...

# Regression modeling

- Forward selection: greedy - add some randomness to the search and go backwards too
- Reduce dimensionality of $X$ by some pre-screening: PCR, PLS, use knowledge of data
- Use regularized regression: ridge, lasso, .....

When $p < n$, we can perform model selection testing "all" subsets of variables to be in the model.
Best model?

- Cross-validation (directly assess the models' predictive performance
- Model selection criteria. AIC: prediction likelihood estimate. BIC: an approximate Bayesian posterior probability measure for the models.

# Model selection

Why is model selection so important?
BIAS-VARIANCE trade-off!

- Complex models: little or no bias in their prediction since they probably include the "true" or a good approximation of the "true" model
- BUT: complex models are very sensitive to data perturbations and if they are more complex than the "true" model they try to make generalizations of noise. Estimation variance is therefor high
- Prediction mean squared error is affected both by BIAS and VARIANCE
- Model selection: about finding the right balance between the two for best predictive performance

# Bias-Variance Trade-off

Depending on the tuning parameter value, a classification rule can be thought of as *local* or *global*.

| local | global |
|---|---|
| use subset of data | use all data |
| flexible | more rigid |
| allow for complex boundaries or models | assume an underlying model for the data distribution |
| example: kNN with small k | example: discriminant analysis (multivariate normal data distribution) |
| example: Local average regression | example: linear regression model |
| need a lot of data to train on | requires less data in general |

# Variable importance

Some examples

- How often is a variable included in the model over many training-validation splits?
- What is the R-squared contribution of the variable (when added first, last or averaged over all subset models that includes it).

# 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 $\pi_c$ is the prior

# 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 $\pi_c$ is the prior probability of class $c$
- ... and many many more.

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

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

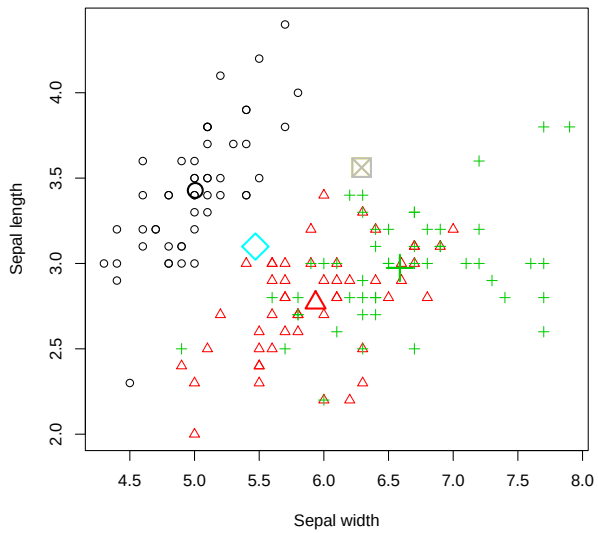
where $d(.,.)$ is the distance between observation location $x$ and the centroid $\mu$. This is usually the euclidean distance

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

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.

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 $\mu_c$ is a p-dimensional vector and $\Sigma_c$ is a p-by-p dimensional covariance matrix.

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 $\pi$s 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.

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.
- $\Sigma_c$: QDA
- $\tilde{\Sigma}_c$ a regularized estimate between $\Sigma_c$, $\Sigma$ and $diag(\Sigma_c)$: RDA

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

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

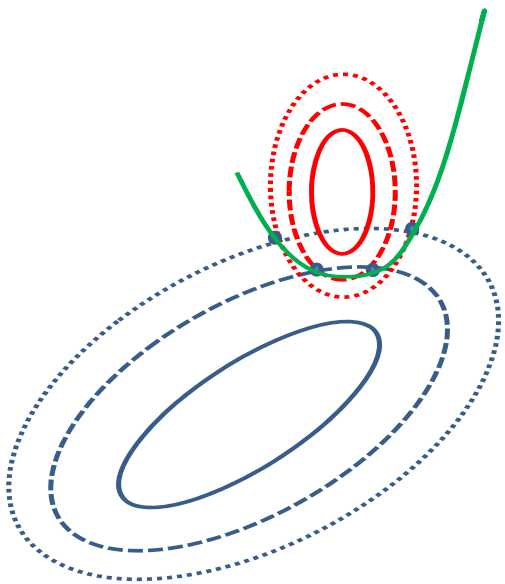
Equivalently, we can write this on a log-scale as

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

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

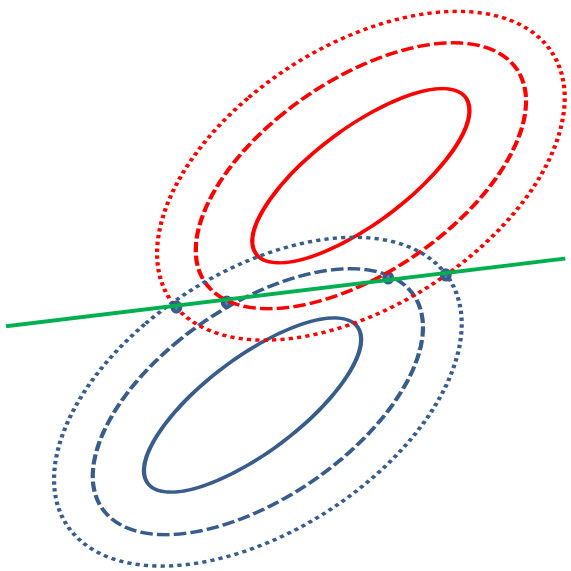
# Building the classifier

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.



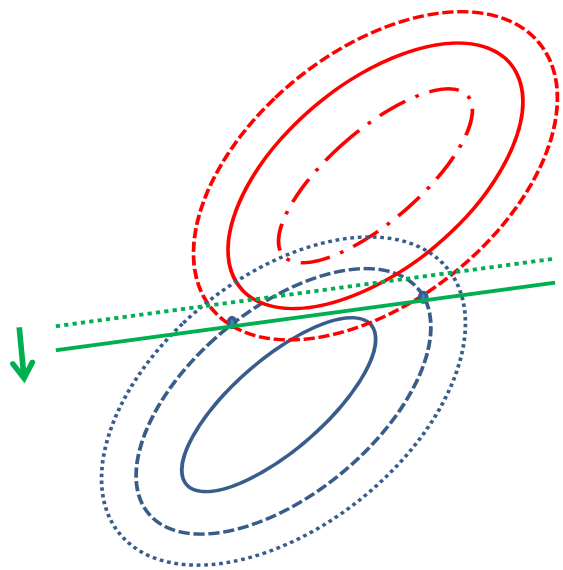
# Building a linear classifier

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, $\pi_{cr}$
- Each component within and between the classes have the same shape, $\Sigma$.

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

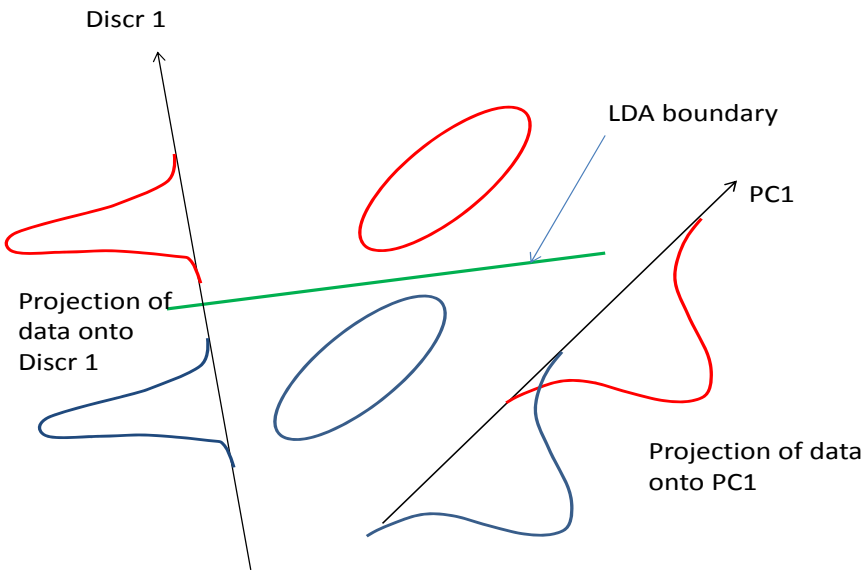
As mentioned in the previous lectures, one problem with LDA stems from the instability of the estimate $\hat{\Sigma}$ when $n$ is small and/or $p$ is large and/or $x$-features are correlated.

# 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 $\Sigma_c$ estimates to common $\Sigma$ 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.

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 $\Sigma$ 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 $\hat{\Sigma}$.

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 $\tilde{x}_2 = d_2^{-1}xu_2$.

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.
- $\Sigma_B$ is the *between-class* variance, the spread of the centroids

We define these as

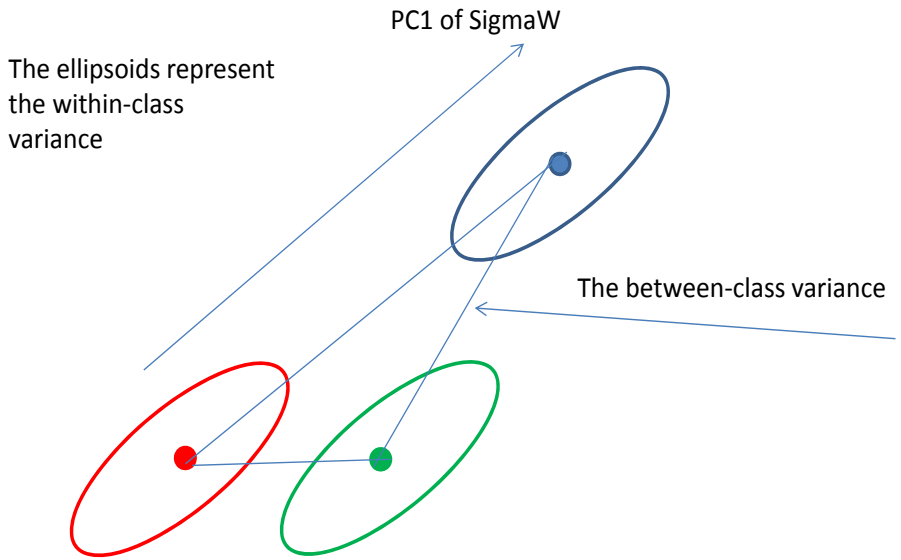
$$\Sigma_W = \sum_{c=1}^{C} \sum_{y_i=c} (x_i - \hat{\mu}_c)(x_i - \hat{\mu}_c)'/(N - C)$$

and

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

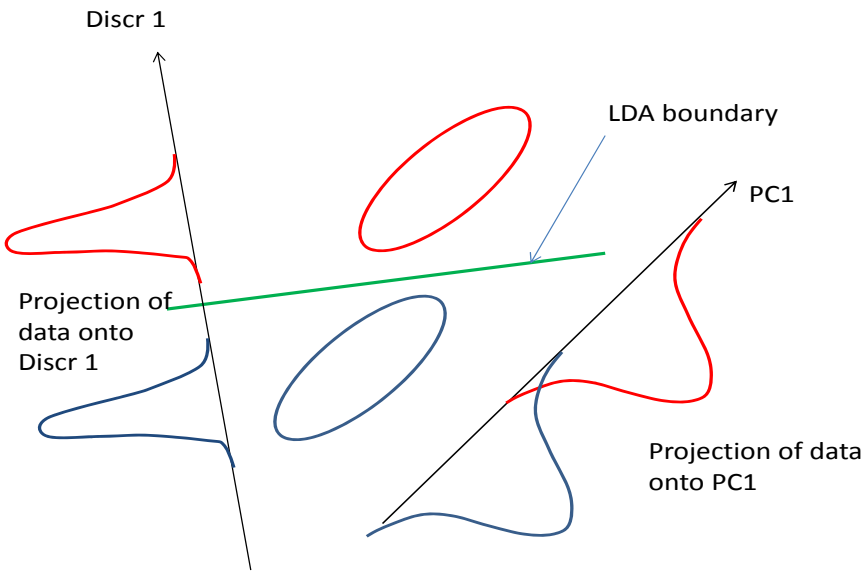
# Fisher's LDA - FLDA

Here's an illustration of the between-class variance $\Sigma_B$ and the within-class variance $\Sigma_W$



# 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

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

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 $\lambda$ 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 $\mu$s with the within-class covariance, $\Sigma_W$: $M^* = M\Sigma_W^{-1/2}$
- 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 $\theta$ is directly proportional to the discriminant vectors $V$ we defined before!

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)

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.

# Ensemble modeling

Using one prediction model, that we select via e.g. cross-validation, means we are betting on there being a clear "winner". Often, when we look and CV-errors (in regression or classification) there are many models that are near equally good. How about combining many models instead?

- Use can use a weighted prediction from the top-K models (e.g top-10). Weights can be equal, or proportional to the relative BIC or CV-error values of the models. This is called *Model averaging*
- You can also use bootstrap to come up with your top-K models. Do repeated training and model selection on many resampled data sets and use the average prediction on the test data. This is called *bagging*
- RandomForest is variant of bagged-CART that further exploits the notion of letting different prediction be aggregated to "wipe out" the noisy part of predictions. Here we use a random subset of predictors as well.
- Note: you can use bagging for any model, not just CART. It's implemented in e.g. `caret` for regression and DA methods.
- When does bagging work? When there is instability in the modeling (e.g. via model selection, or because $p$ is large). Bagging reduces estimation variance!

# Regularized regression

$p > n$ problem: use penalized least squares

$$\frac{1}{2}||y - X\beta||^2 + \lambda J(\beta)$$

- As you saw previous lectures the penalty $J$ can be chosen in many ways
- $J(\beta) = ||\beta||_2^2$ (L2) gives us the ridge regression estimates.
- $J(\beta) = ||\beta||_1$ (L1) gives us the lasso estimates
- Both are biased, lasso with a constant bias $\lambda$ for non-zero estimates whereas ridge has an increasing bias with coefficient magnitude.
- Adaptive lasso uses a penalty that is coefficient specific $J(\beta) = \sum_{j=1}^{p} |\beta_j| w_j$ where $w_j = \frac{1}{|\beta_{j,init}|^\gamma}$ and this reduces the bias if $\beta_{init}$ is chosen well (a $\sqrt{n}$-consistent estimator)

# Regularized regression

$p > n$ problem: use penalized least squares

$$\frac{1}{2}||y - X\beta||^2 + \lambda J(\beta)$$

- There were more complex $J()$ we could consider
- *group-lasso* $J(\beta) = \sum_g^G ||\beta_g||_2$ penalizes a group mean for the coefficients which has the effect of selecting a group of coefficients to be "in" or "out" in the model
- If you add the L1-penalty $||\beta||_1$ to the group penalty you get *sparse group-lasso* which favors groups to enter the model but "cleans up" the coefficients in the group that are not contributing.
- *Elastic net* uses a weighted combination of the L2 and L1 penalties. This has the effect of keeping correlated variables together in the model. You don't have to define the groups.

# Elastic net

$$\frac{1}{2}||y - X\beta||^2 + (1-\alpha)\lambda||\beta||_2^2 + \alpha\lambda||\beta||_1$$

- Zou and Hastie proposed the elastic net in 2005
- The motivation was the observation that if variables are correlated, lasso tends to include only one variable from the group
- Furthermore, it was seen that lasso performs poorly compared to ridge regression when there are correlated predictors present.
- Why not group lasso? Sometimes it is not so easy to determine the groups - another tuning process where we cluster variables first.

# Elastic net

$$\frac{1}{2}||y - X\beta||^2 + (1-\alpha)\lambda||\beta||_2^2 + \alpha\lambda||\beta||_1$$

- The bigger $\alpha$ is the more grouping of variables you get
- How much grouping depends on the pairwise correlations between variables in comparison to the penalty $\lambda(1-\alpha)$.

When we have categorical outcome data we can still use penalized fitting strategies.
The objective function will now be the negative log-likelihood of the data plus the penalty.

# High-dimensional inference

- L1-penalized modeling has become enormously popular for high-dimensional problems
- We get model selection, fairly good predictions and as saw above, good point estimates
- But when we do low-dimensional modeling we usually don't feel very satisfied with just point estimates
- We want confidence intervals and p-values!

# High-dimensional inference

- What are the obstacles for obtaining p-values and confidence intervals?
- Highly non-standard setting when $p > n$
- the distribution of lasso-solutions, by construction, has a point-mass at 0 and this makes bootstrapping to get standard error estimates difficult

# Sample-splitting

Wasserman and Roeder (2009) proposed the following approach to obtain p-values

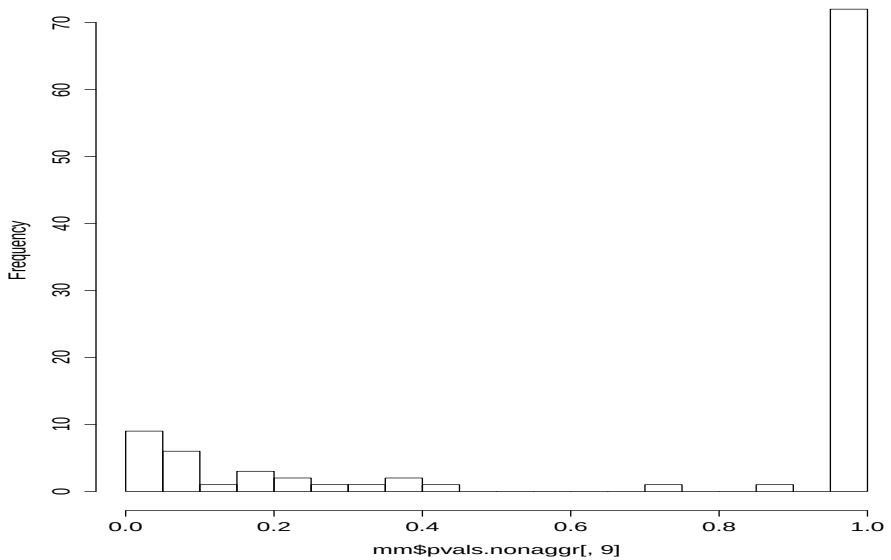
- Split the data in two sets
- Use set 1 to perform modelselection via e.g. lasso
- Use set 2 to evaluate p-values for the non-zero coefficients. This is done by running LS using only the selected variables in the model.
- For the variables not selected in set 1, set p-value to 1.

The p-values are valid because we didn't reuse the same data for selection and p-value computation.
Moreover, if we want to compute adjusted p-values that take into account multiple testing we only have to correct by the selected set of variables, not all $p$.

# Sample-splitting

Drawback with the procedure

- Sensitive to the split so the pvalues are not reproducible
- "p-value lottery"
- Different splits leads to widely different p-values!



# Multi sample-splitting

To overcome the p-value lottery we perform several random splits of data (Meinhausen et al, 2009)

- Repeat $B$ times: split data into set 1 and set 2
- Use set 1 for selection of variables
- Use set 2 to compute p-values
- Aggregate the p-values

Hm? How to we combine $B$ p-values (like those from the histogram above) to one final p-value?

# Multi sample-splitting

The p-value estimates are not independent because the data splits overlap.

- We can use the median p-value
- Or any other quantile
- Search for the best quantile
- Implemented in package `hdi()`

# De-sparsified lasso

There's been a lot of work in the last 2-3 years on the p-value and confidence interval problem of sparse estimators.
Zhang and Zhang (2014) proposed the de-sparsified lasso to come up with p-values in a high-dimensional setting. Another proposal by Buhlmann (2013) uses a bias-corrected ridge estimate.
The sampling distribution of the bias-corrected estimates do not have a point-mass at zero and one arrives at an asymptotic normal distribution that can be used for inference.

# Correlated variables

In practice, we often have highly correlated variables in our data sets. This was the motivation for group selection in elastic net or group lasso.
When we have correlated variables this translates to higher estimation variance within the group, wider confidence intervals and lower power of detection.

- Group testing is one solution
- We can group the variables together based on their pairwise correlations, e.g. via hierarchical clustering
- We can then compute p-values for each group.
- How do we we generate group-p-values?
- In de-sparsified lasso and ridge we adjust the individual p-values by the number of tests performed ($p$) and the then use the minimum adjusted p-value within the group for group decisions.

# Correlated variables

Meinhausen (2013) proposes a multi-split testing of groups as follows.

- We use multi-sample splitting to construct confidence intervals for the l1-norm of a group.
- If the lower bound of this confidence interval is larger than 0, we reject the null-hypothesis for this group.
- `hdi()` package illustrates the group tests with a hierarchical tree (see demo)