Summary: Discriminant Analysis

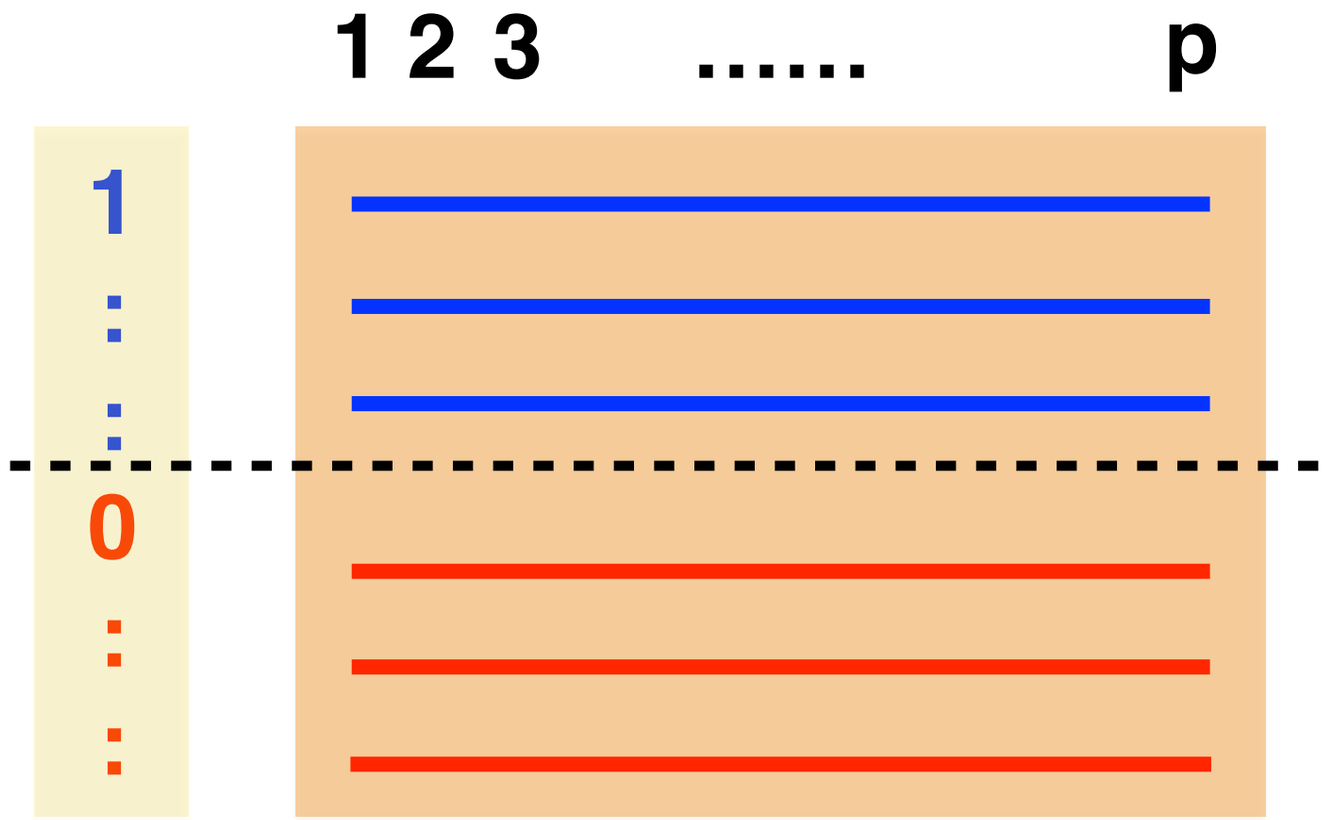
1. **Limitation and advantages of LDA/NB**
2. **Suggest to use screening procedures to reduce p**
3. **In general, do not recommend QDA; check RDA**

$$P(x, y) = P(Y=y \mid X=x)\, P(X=x)$$
$$= P(X=x \mid Y=y)\, P(Y=y)$$

Joint dist | **Conditions of XIY** | **Marginal of Y**

Dist of p-dim X given Y=k: **QDA, LDA (FDA), NB**

Data

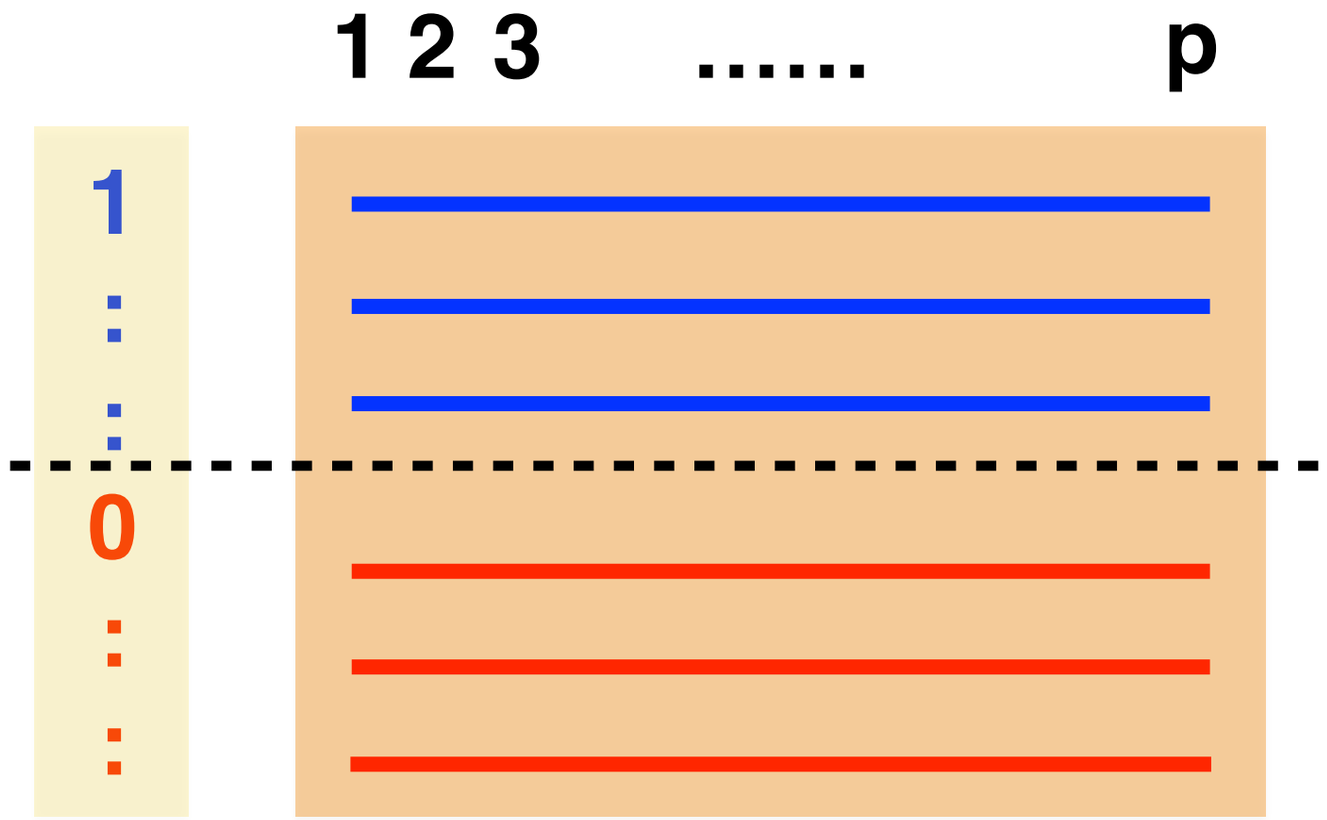


LDA in High-dim

Parameters need to estimate

mu_1, mu_2, Sigma, pi_1

What we need is **inverse of Sigma**



LDA in High-dim

Parameters need to estimate

mu_1, mu_2, Sigma, pi_1

What we need is **inverse of Sigma**

Each element of mu_k's and Sigma can be reliably estimated with a reasonable sample size. However, Sigma_inverse is **error-prone**.

How accurately can we estimate the first PC direction?

```
set.seed(123)
n=500;
p.seq=seq(10, 300, by=10)
m = length(p.seq)
mycor = rep(0, m)
for(i in 1:m){
  p = p.seq[i]
  X = matrix(rnorm(n*p), n, p)
  X[, 1] = X[, 1]*sqrt(2)
  tmp = cov(X)
  pc1 = svd(tmp)$u[,1]
  mycor[i] = pc1[1]
}

plot(p.seq, abs(mycor),
  ylab="Correlation",
  xlab="Dimension" )
```

X_1, …, X_n are iid N(0, Sigma)

True Sigma = diag(2, 1, …, 1)
True PC1 = c(1, 0, …, 0)

Plot the correlation of True PC1 and the estimated one.

How accurately can we estimate the first PC direction?

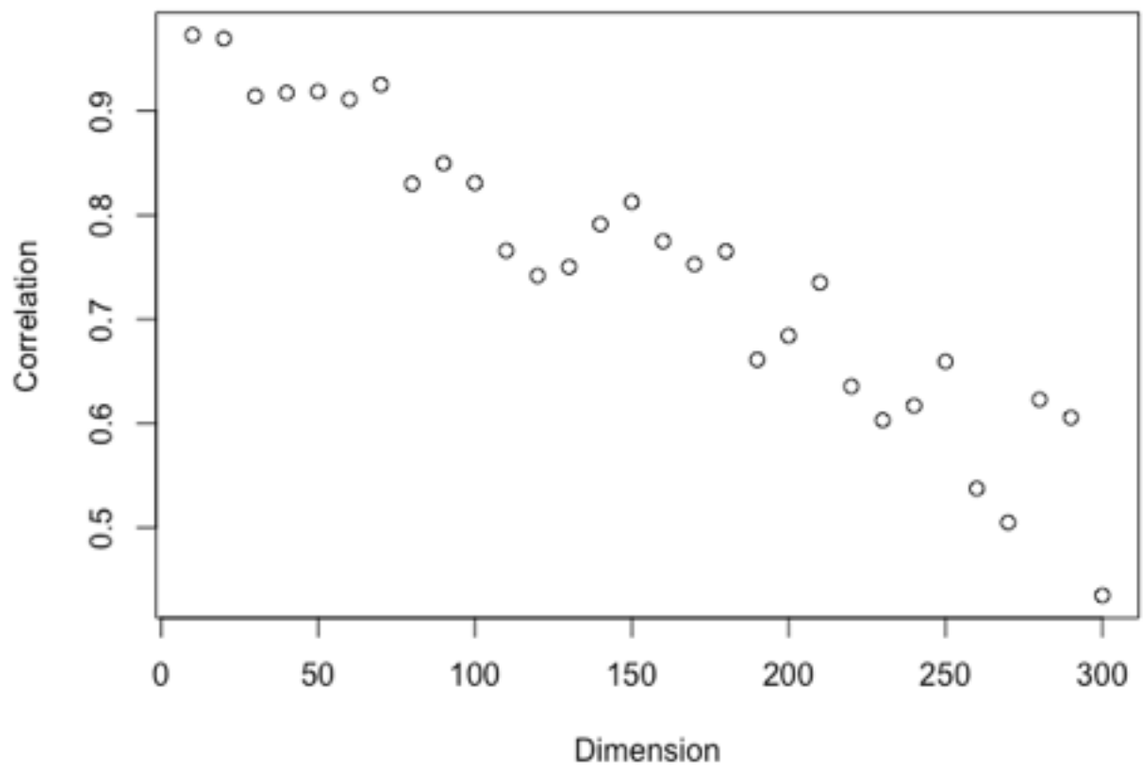
```
set.seed(123)
n=500;
p.seq=seq(10, 300, by=10)
m = length(p.seq)
mycor = rep(0, m)
for(i in 1:m){
  p = p.seq[i]
  X = matrix(rnorm(n*p), n, p)
  X[, 1] = X[, 1]*sqrt(2)
  tmp = cov(X)
  pc1 = svd(tmp)$u[,1]
  mycor[i] = pc1[1]
}

plot(p.seq, abs(mycor),
  ylab="Correlation",
  xlab="Dimension" )
```

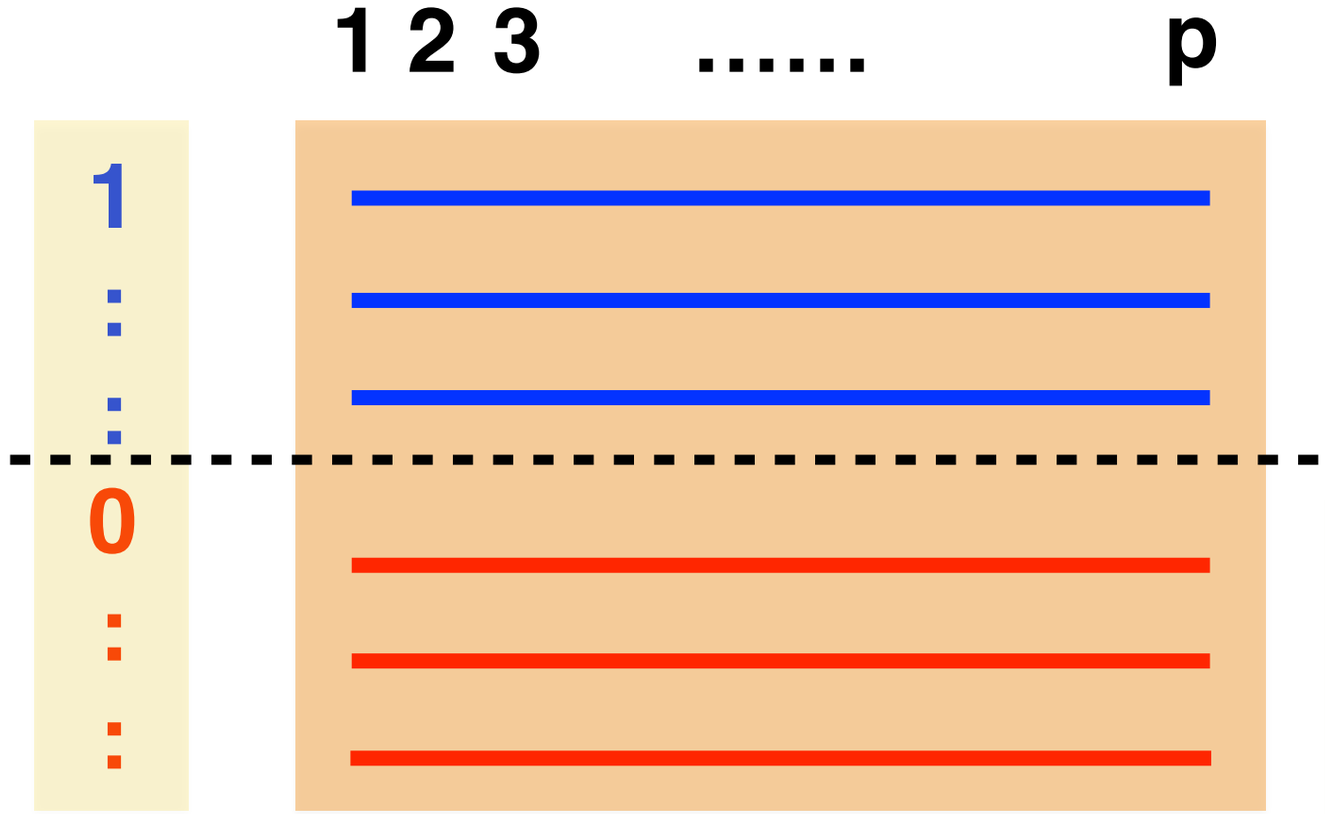
X_1, …, X_n are iid N(0, Sigma)

True Sigma = diag(2, 1, …, 1)
True PC1 = c(1, 0, …, 0)

Plot the correlation of True PC1 and the estimated one.



Data



Wait ~~~ We do not need the whole Sigma_inverse matrix to be accurate.

What we need is the accuracy of its inner product with (mu1 - mu2)

LDA in High-dim

Parameters need to estimate
mu1, mu2, Sigma, pi_1
What we need is **inverse of Sigma**

Each element of mu_k's and Sigma can be reliably estimated with a reasonable sample size. However, Sigma_inverse is **error-prone**.

For example, for binary LDA with discriminant function (1):

$$d_k(\mathbf{x}) = -2\mathbf{x}^t\Sigma^{-1}\boldsymbol{\mu}_k + \boldsymbol{\mu}_k^T\Sigma^{-1}\boldsymbol{\mu}_k - 2\log\pi_k$$

What matters is the decision boundary which is a linear function has $(p+1)$ parameters:

$$d_1(\mathbf{x}) - d_2(\mathbf{x}) = -2\mathbf{x}^t\Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) + \beta_0 = \mathbf{x}^t\boldsymbol{\beta} + \beta_0.$$

However, we estimate $(\boldsymbol{\beta}, \beta_0)$ by learning a much larger collection of parameters such as Σ, $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$ and π_1.

- Next we'll discuss how to directly learn $P(Y = k|X = \mathbf{x})$ (e.g., logistic regression, tree models) or directly learn the decision boundary (e.g., SVM).

Should We Worry About the Normality Assumption?

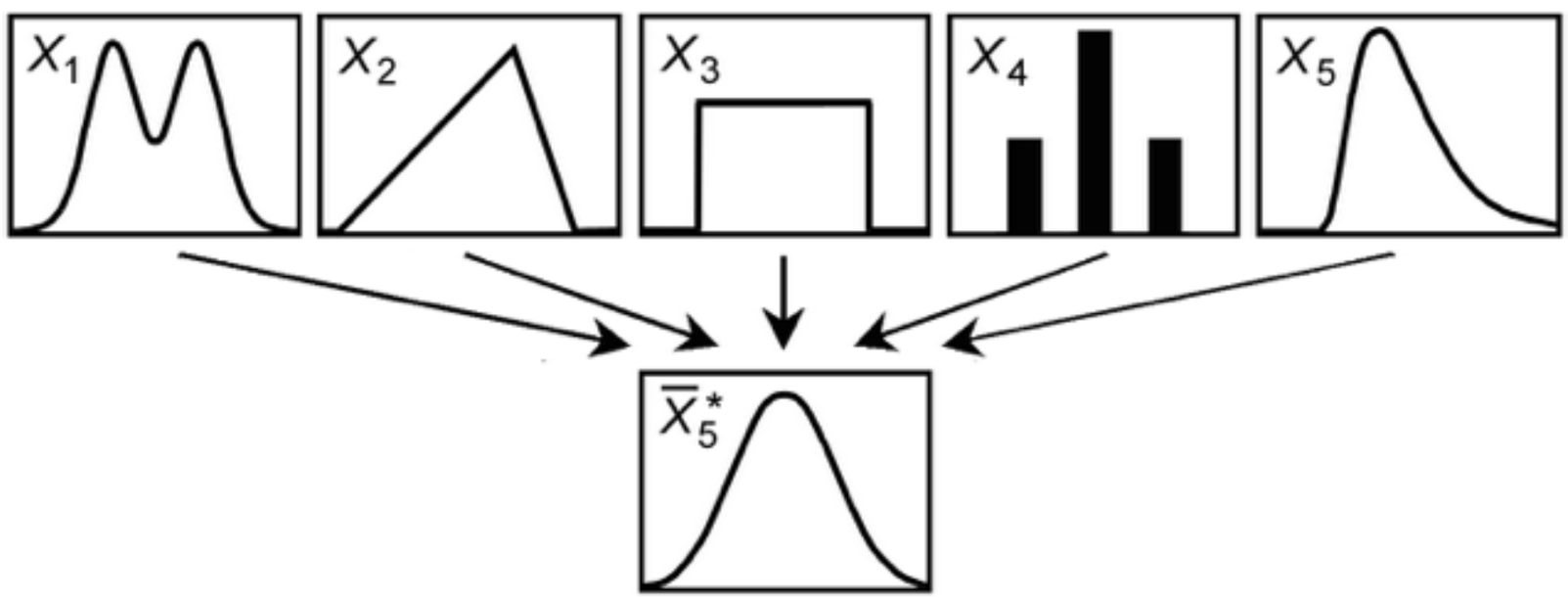
The Annals of Statistics
1984, Vol. 12, No. 3, 793–815

ASYMPTOTICS OF GRAPHICAL PROJECTION PURSUIT

BY PERSI DIACONIS[1] AND DAVID FREEDMAN[2]

Stanford University and University of California, Berkeley

Mathematical tools are developed for describing low-dimensional projections of high-dimensional data. Theorems are given to show that under suitable conditions, most projections are approximately Gaussian.



Summary: Discriminant Analysis

In Discriminant Analysis (DA), we estimate the joint

$$P(X = \mathbf{x}, Y = k) = P(X = \mathbf{x}|Y = k) \times P(Y = k),$$

and then obtain $P(Y = k|X = \mathbf{x})$.

Can Naturally incorporate unlabelled data.

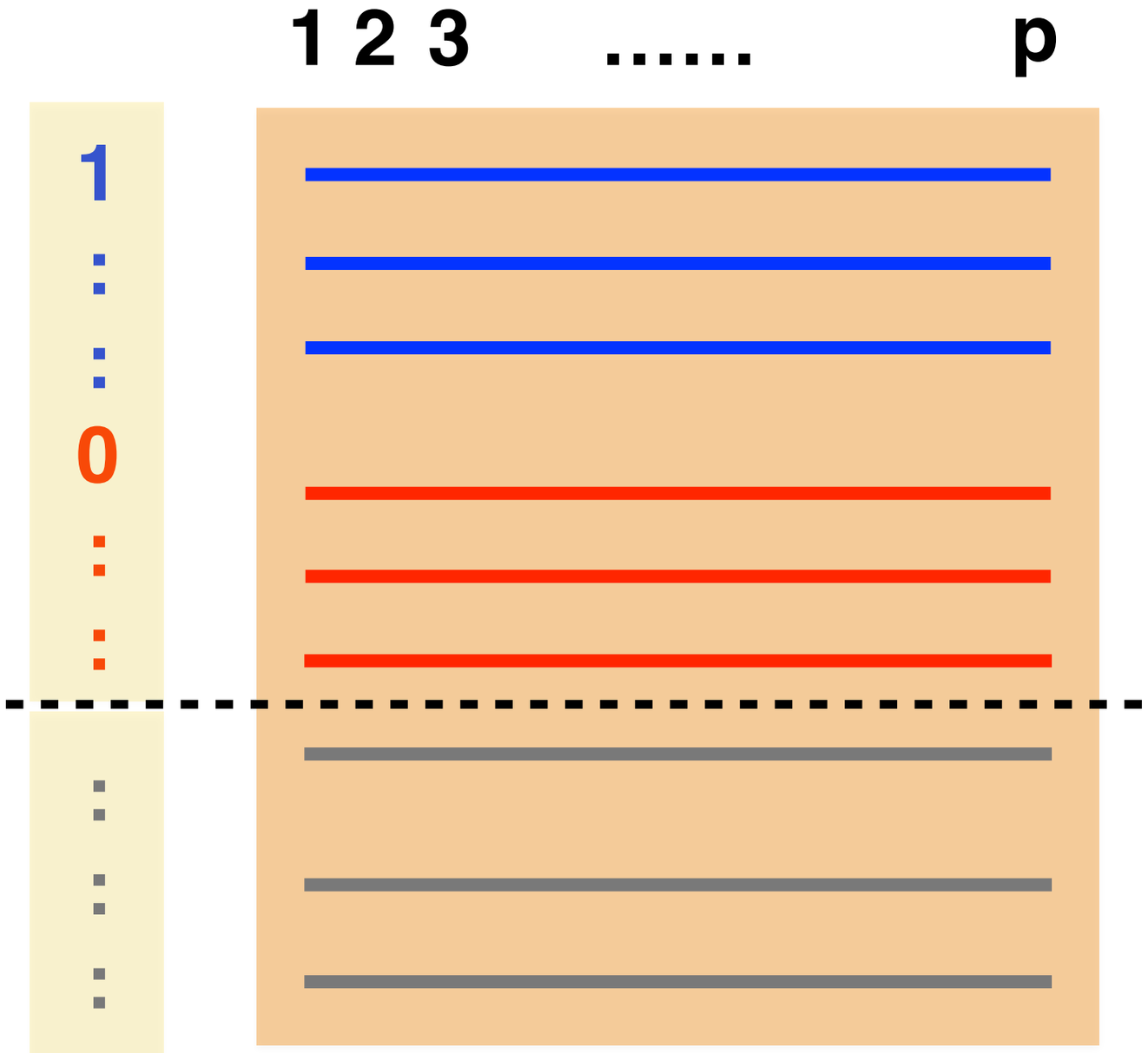
DA is conceptually simple and works for some low-dimensional problems,

but not an effective way of building classifiers.

P(x, y) = P(Y=y | X=x) P(X=x)
= P(X=x | Y=y) P(Y=y)

Joint dist **Conditions of XIY** **Marginal of Y**

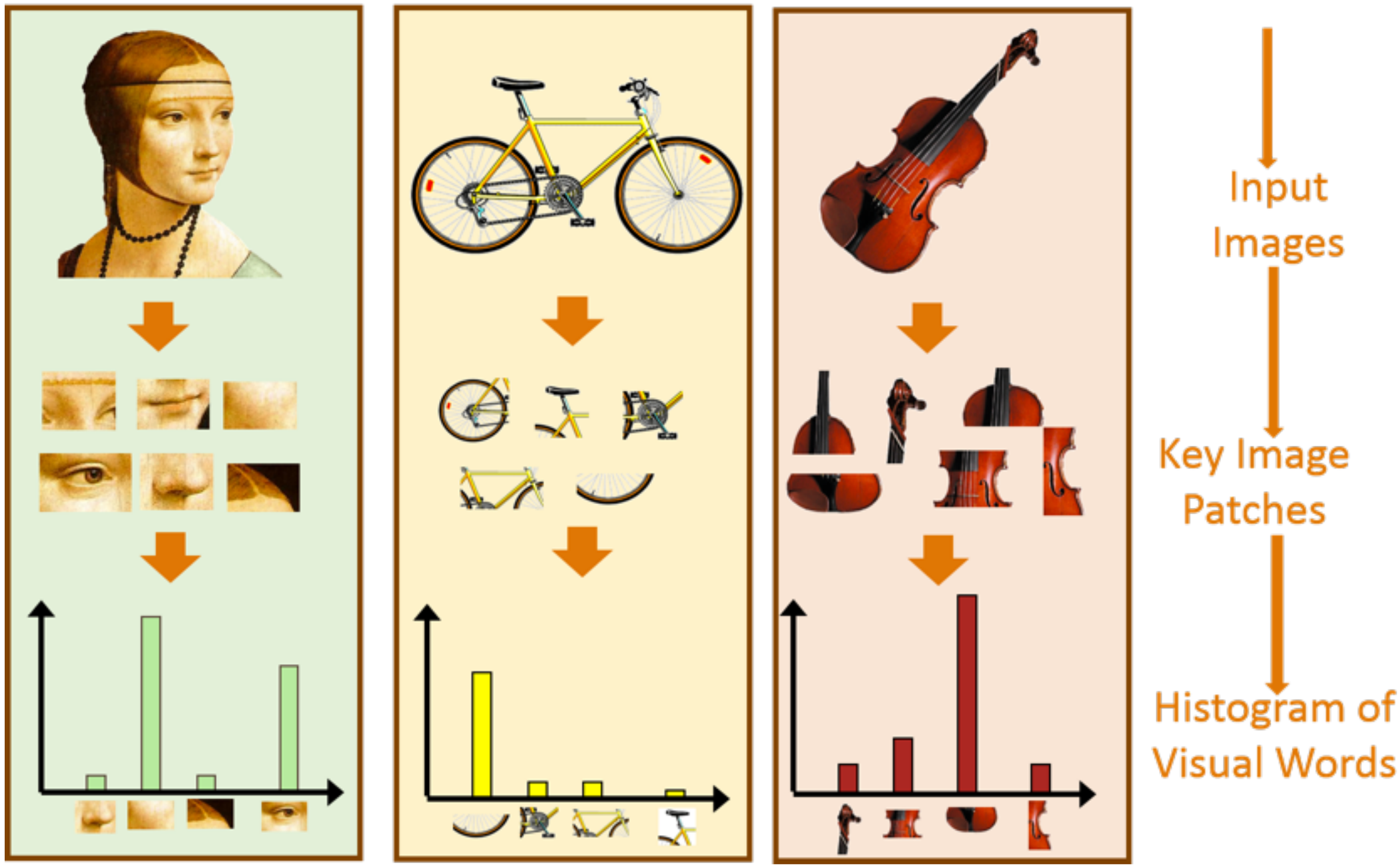
Dist of p-dim X given Y=k: **QDA, LDA (FDA), NB**



The underlying model assumption for (X,Y) is the same.

1. LDA: Y is given
2. EM for Mixture Model: Y is unknown latent variable
3. For **semi-supervised learning**, we can combine these two.

Image Classification: LDA+LDA



LDA: Latent Dirichlet Allocation

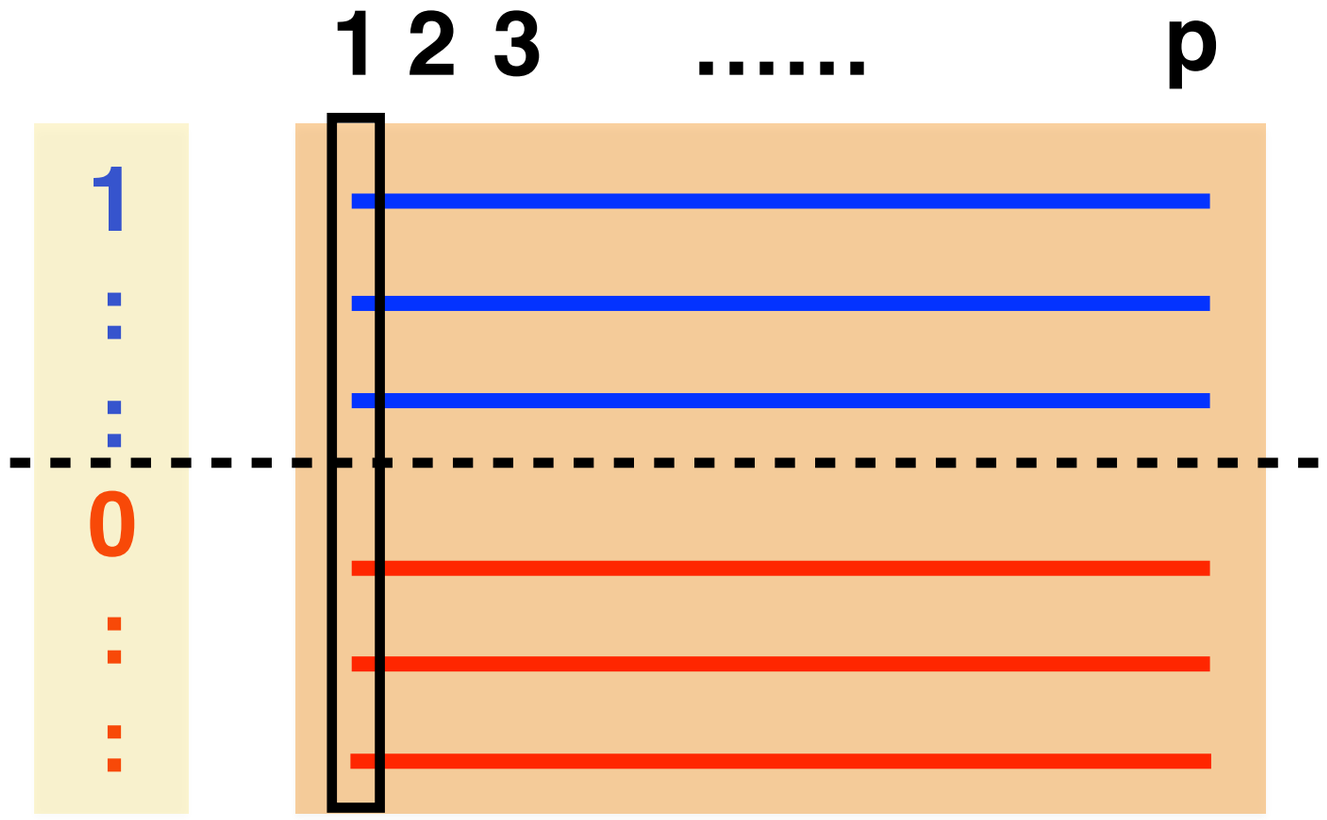
LDA: Linear Discriminant Analysis

Image Classification: LDA+LDA

Classify the three patterns:
LDA+LDA should work well; no need to use DL

LDA: Latent Dirichlet Allocation
LDA: Linear Discriminant Analysis

Data




```
for j=1:p
  Group 1:
    x_[1,j]
    x_[2,j]
    ...
    x_[n1, j]

  Group 0:
    x_[n1+1,j]
    x_[n1+2,j]
    ...
    x_[n1+n0,j]
```

LDA/NB in High-dim

Pre-screening variables to reduce p,
e.g., **two-sample t-test, or its variants**

Rank the p-values for the p features, and drop features with large p-values.

Regularized Discriminant Analysis

RDA uses the following regularized covariance matrix

Group Sigma = Average of three

$$\hat{\Sigma}_k(\lambda,\gamma) = (1-\gamma)\hat{\Sigma}_k(\lambda) + \gamma\frac{1}{p}\text{tr}\left[\hat{\Sigma}_k(\lambda)\right]\mathbf{I}_p,$$

$$\hat{\Sigma}_k(\lambda) \equiv (1-\lambda)\hat{\Sigma}_k + \lambda\hat{\Sigma},$$

with $\lambda, \gamma \in [0,1]$ [a] Large values indicate higher degrees of regularization.

- $(\gamma = 0, \lambda = 0)$: QDA (individual cov for each class).
- $(\gamma = 0, \lambda = 1)$: LDA (shared cov matrix).
- $(\gamma = 1, \lambda = 0)$: Variables are conditionally independent with equal class-specific variance; similar to Naive Bayes.
- $(\gamma = 1, \lambda = 1)$: Nearest centroid (objects are assigned to group with nearest mean with euclidean distance).

Factors: Ordered or Unordered?

```
> X=rep(c("A", "B", "C"),
    times=c(2, 3, 1))
> X = as.factor(X)
> y = rnorm(6)
> fit1 = lm(y~X)
> model.matrix(fit1)
  (Intercept) XB XC
1           1  0  0
2           1  0  0
3           1  1  0
4           1  1  0
5           1  1  0
6           1  0  1
attr(,"assign")
[1] 0 1 1
attr(,"contrasts")
attr(,"contrasts")$X
[1] "contr.treatment"
```

```
> fit2 = lm(y~X-1)
> model.matrix(fit2)
  XA XB XC
1  1  0  0
2  1  0  0
3  0  1  0
4  0  1  0
5  0  1  0
6  0  0  1
attr(,"assign")
[1] 1 1 1
attr(,"contrasts")
attr(,"contrasts")$X
[1] "contr.treatment"
```

Factors: Ordered or Unordered?

```
> attr(X, "contrasts") =
                contr.sum(3)
> contrasts(X)
  [,1] [,2]
1    1    0
2    0    1
3   -1   -1
> fit3 = lm(y~X)
> model.matrix(fit3)
  (Intercept) X1 X2
1           1  1  0
2           1  1  0
3           1  0  1
4           1  0  1
5           1  0  1
6           1 -1 -1
```

```
> attr(X, "contrasts") =
                contr.poly(3)
> contrasts(X)
           .L     .Q
[1,] -0.707  0.408
[2,]  0.000 -0.816
[3,]  0.707  0.408
> fit4 = lm(y~X)
> model.matrix(fit4)
  (Intercept)    X.L    X.Q
1           1 -0.707  0.408
2           1 -0.707  0.408
3           1  0.000 -0.816
4           1  0.000 -0.816
5           1  0.000 -0.816
6           1  0.707  0.408
```

Factors: Ordered or Unordered?

```
> XX = factor(X, ordered=TRUE)
> contrasts(XX)
           .L      .Q
[1,] -0.707  0.408
[2,]  0.000 -0.816
[3,]  0.707  0.408
> fit5 = lm(y ~ XX)
> model.matrix(fit5)
  (Intercept)    XX.L    XX.Q
1           1 -0.707  0.408
2           1 -0.707  0.408
3           1  0.000 -0.816
4           1  0.000 -0.816
5           1  0.000 -0.816
6           1  0.707  0.408
attr(,"assign")
[1] 0 1 1
attr(,"contrasts")
attr(,"contrasts")$XX
[1] "contr.poly"
```

```
> attr(X, "contrasts") =
                    contr.poly(3)
> contrasts(X)
           .L      .Q
[1,] -0.707  0.408
[2,]  0.000 -0.816
[3,]  0.707  0.408
> fit4 = lm(y~X)
> model.matrix(fit4)
  (Intercept)     X.L     X.Q
1           1 -0.707  0.408
2           1 -0.707  0.408
3           1  0.000 -0.816
4           1  0.000 -0.816
5           1  0.000 -0.816
6           1  0.707  0.408
```

Factors: Ordered or Unordered?

```
> myout = c(summary(fit1)$sigma, fit1$coef)
> myout = rbind(myout, c(summary(fit2)$sigma, fit2$coef))
> myout = rbind(myout, c(summary(fit3)$sigma, fit3$coef))
> myout = rbind(myout, c(summary(fit4)$sigma, fit4$coef))
> myout = rbind(myout, c(summary(fit5)$sigma, fit5$coef))

> myout
                  (Intercept)          XB          XC
myout 0.4140462  0.007923035 -0.8786437 -0.8376191 treatment
      0.4140462  0.007923035 -0.8707206 -0.8296961 naive
      0.4140462 -0.564164552  0.5720876 -0.3065561 sum
      0.4140462 -0.564164552 -0.5922861  0.3754530 poly
      0.4140462 -0.564164552 -0.5922861  0.3754530 ordered
```

Ordered or unordered: does it make any difference?

- **No difference for linear model (of course, coefficients are different, but prediction is the same)**
- **May lead to different variable selection result**