Plug-in Classification

- Consider the 0-1 loss and the risk:

\[
\mathbb{E}\left[ L(Y, f(X)) \mid X = x \right] = \sum_{k=1}^{K} L(k, f(x)) \mathbb{P}(Y = k \mid X = x)
\]

The Bayes classifier provides a solution that minimizes the risk:

\[
f_{\text{Bayes}}(x) = \arg \max_{k=1,\ldots,K} \pi_k g_k(x).
\]

- We know neither the conditional density \( g_k \) nor the class probability \( \pi_k \)!
- The **plug-in classifier** chooses the class

\[
f(x) = \arg \max_{k=1,\ldots,K} \hat{\pi}_k \hat{g}_k(x),
\]

where we plugged in

- estimates \( \hat{\pi}_k \) of \( \pi_k \) and \( k = 1, \ldots, K \) and
- estimates \( \hat{g}_k(x) \) of conditional densities,

**Linear Discriminant Analysis** is an example of plug-in classification.
Summary: **Linear Discriminant Analysis**

- **LDA**: a plug-in classifier assuming multivariate normal conditional density $g_k(x) = g_k(x|\mu_k, \Sigma)$ for each class $k$ sharing the same covariance $\Sigma$:

  $$X|Y = k \sim \mathcal{N}(\mu_k, \Sigma),$$

  $$g_k(x|\mu_k, \Sigma) = (2\pi)^{-p/2} |\Sigma|^{-1/2} \exp \left( -\frac{1}{2} (x - \mu_k)^\top \Sigma^{-1} (x - \mu_k) \right).$$

- LDA minimizes the squared **Mahalanobis distance** between $x$ and $\hat{\mu}_k$, offset by a term depending on the estimated class proportion $\hat{\pi}_k$:

  $$f_{\text{LDA}}(x) = \arg\max_{k \in \{1, \ldots, K\}} \log \hat{\pi}_k g_k(x|\hat{\mu}_k, \hat{\Sigma})$$

  $$= \arg\max_{k \in \{1, \ldots, K\}} \left( \log \hat{\pi}_k - \frac{1}{2} \hat{\mu}_k^\top \hat{\Sigma}^{-1} \hat{\mu}_k \right) + \left( \hat{\Sigma}^{-1} \hat{\mu}_k \right)^\top x$$

  terms depending on $k$ linear in $x$

  $$= \arg\min_{k \in \{1, \ldots, K\}} \frac{1}{2} (x - \hat{\mu}_k)^\top \hat{\Sigma}^{-1} (x - \hat{\mu}_k) - \log \hat{\pi}_k.$$  

  squared Mahalanobis distance
LDA projections

Figure by R. Gutierrez-Osuna
LDA vs PCA projections

LDA separates the groups better.
Fisherfaces

Eigenfaces vs. Fisherfaces, Belhumeur et al. 1997

Conditional densities with different covariances

Given training data with \( K \) classes, assume a parametric form for conditional density \( g_k(x) \), where for each class

\[
X | Y = k \sim \mathcal{N}(\mu_k, \Sigma_k),
\]
i.e., instead of assuming that every class has a different mean \( \mu_k \) with the same covariance matrix \( \Sigma \) (LDA), we now allow each class to have its own covariance matrix.

Considering \( \log \pi_k g_k(x) \) as before,

\[
\log \pi_k g_k(x) = \text{const} + \log(\pi_k) - \frac{1}{2} \left( \log |\Sigma_k| + (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) \right)
\]

\[
= \text{const} + \log(\pi_k) - \frac{1}{2} \left( \log |\Sigma_k| + \mu_k^T \Sigma_k^{-1} \mu_k \right) + \mu_k^T \Sigma_k^{-1} x - \frac{1}{2} x^T \Sigma_k^{-1} x
\]

\[
= a_k + b_k^T x + x^T c_k x.
\]

A quadratic discriminant function instead of linear.
Quadratic decision boundaries

Again, by considering that we choose class $k$ over $k'$,

$$a_k + b_k^T x + x^T c_k x - (a_{k'} + b_{k'}^T x + x^T c_{k'} x)$$

$$= a_\star + b_\star^T x + x^T c_\star x > 0$$

we see that the decision boundaries of the Bayes Classifier are quadratic surfaces.

- The plug-in Bayes Classifier under these assumptions is known as the **Quadratic Discriminant Analysis** (QDA) Classifier.
QDA

LDA classifier:

\[
 f_{\text{LDA}}(x) = \arg\min_{k \in \{1, \ldots, K\}} \left\{ (x - \hat{\mu}_k)^T \hat{\Sigma}_k^{-1} (x - \hat{\mu}_k) - 2 \log(\hat{\pi}_k) \right\}
\]

QDA classifier:

\[
 f_{\text{QDA}}(x) = \arg\min_{k \in \{1, \ldots, K\}} \left\{ (x - \hat{\mu}_k)^T \hat{\Sigma}_k^{-1} (x - \hat{\mu}_k) - 2 \log(\hat{\pi}_k) + \log(|\hat{\Sigma}_k|) \right\}
\]

for each point \( x \in \mathcal{X} \) where the plug-in estimate \( \hat{\mu}_k \) is as before and \( \hat{\Sigma}_k \) is (in contrast to LDA) estimated for each class \( k = 1, \ldots, K \) separately:

\[
 \hat{\Sigma}_k = \frac{1}{n_k} \sum_{j: y_j = k} (x_j - \hat{\mu}_k)(x_j - \hat{\mu}_k)^T.
\]
Computing and plotting the QDA boundaries.

```r
## fit QDA
iris.qda <- qda(x=iris.data, grouping=ct)

## create a grid for our plotting surface
x <- seq(-6, 6, 0.02)
y <- seq(-4, 4, 0.02)
z <- as.matrix(expand.grid(x, y), 0)
m <- length(x)
n <- length(y)

iris.qdp <- predict(iris.qda, z)$class
contour(x, y, matrix(iris.qdp, m, n),
        levels = c(1.5, 2.5), add = TRUE, d = FALSE, lty = 2)
```
Iris example: QDA boundaries
Iris example: QDA boundaries
LDA or QDA?

- Having seen both LDA and QDA in action, it is natural to ask which is the “better” classifier.
- If the covariances of different classes are very distinct, QDA will probably have an advantage over LDA.
- Parametric models are only ever approximations to the real world, allowing more flexible decision boundaries (QDA) may seem like a good idea. However, there is a price to pay in terms of increased variance and potential overfitting.
Regularized Discriminant Analysis

In the case where data is scarce, to fit

- LDA, need to estimate $K \times p + p \times p$ parameters
- QDA, need to estimate $K \times p + K \times p \times p$ parameters.

Using LDA allows us to better estimate the covariance matrix $\Sigma$. Though QDA allows more flexible decision boundaries, the estimates of the $K$ covariance matrices $\Sigma_k$ are more variable.

RDA combines the strengths of both classifiers by regularizing each covariance matrix $\Sigma_k$ in QDA to the single one $\Sigma$ in LDA

$$
\Sigma_k(\alpha) = \alpha \Sigma_k + (1 - \alpha) \Sigma \quad \text{for some } \alpha \in [0, 1].
$$

This introduces a new parameter $\alpha$ and allows for a continuum of models between LDA and QDA to be used. Can be selected by Cross-Validation for example.
Logistic regression
In LDA and QDA, we estimate $p(x|y)$, but for classification we are mainly interested in $p(y|x)$.

Why not estimate that directly? Logistic regression\(^1\) is a popular way of doing this.

\(^1\)Despite the name “regression”, we are using it for classification!
Logistic regression

- One of the most popular methods for classification
- Linear model on the probabilities
- Dates back to work on population growth curves by Verhulst [1838, 1845, 1847]
- Statistical use for classification dates to Cox [1960s]
- Independently discovered as the perceptron in machine learning [Rosenblatt 1957]
- Main example of “discriminative” as opposed to “generative” learning
- Naïve approach to classification: we could do linear regression assigning specific values to each class. Logistic regression refines this idea and provides a more suitable model.
Logistic regression

- Statistical perspective: consider $\mathcal{Y} = \{0, 1\}$. Generalised linear model with Bernoulli likelihood and logit link:

$$Y | X = x, a, b \sim \text{Bernoulli}\left(s(a + b^\top x)\right)$$

$$s(a + b^\top x) = \frac{1}{1 + \exp(- (a + b^\top x))}.$$  

- ML perspective: a discriminative classifier. Consider binary classification with $\mathcal{Y} = \{+1, -1\}$. Logistic regression uses a parametric model on the conditional $Y | X$, not the joint distribution of $(X, Y)$:

$$p(Y = y | X = x; a, b) = \frac{1}{1 + \exp(-y(a + b^\top x))}.$$
Prediction Using Logistic Regression
Consider using LDA for binary classification with $\mathcal{Y} = \{+1, -1\}$. Predictions are based on linear decision boundary:

$$\hat{y}_{LDA}(x) = \text{sign}\left\{ \log \hat{\pi}_{+1}g_{+1}(x|\hat{\mu}_{+1}, \hat{\Sigma}) - \log \hat{\pi}_{-1}g_{-1}(x|\hat{\mu}_{-1}, \hat{\Sigma}) \right\}$$

$$= \text{sign}\left\{ a + b^\top x \right\}$$

for $a$ and $b$ depending on fitted parameters $\hat{\theta} = (\hat{\pi}_{+1}, \hat{\pi}_{-1}, \hat{\mu}_{+1}, \hat{\mu}_{-1}, \Sigma)$.

Quantity $a + b^\top x$ can be viewed as a soft classification rule. Indeed, it is modelling the difference between the log-discriminant functions, or equivalently, the log-odds ratio:

$$a + b^\top x = \log \frac{p(Y = +1|X = x; \hat{\theta})}{p(Y = -1|X = x; \hat{\theta})}.$$ 

$f(x) = a + b^\top x$ corresponds to the “confidence of predictions” and loss can be measured as a function of this confidence:

- exponential loss: $L(y, f(x)) = e^{-yf(x)}$,
- log-loss: $L(y, f(x)) = \log(1 + e^{-yf(x)})$,
- hinge loss: $L(y, f(x)) = \max\{1 - yf(x), 0\}$. 


Linearity of log-odds and logistic function

- $a + b^\top x$ models the **log-odds ratio**:

$$\log \frac{p(Y = +1|X = x; a, b)}{p(Y = -1|X = x; a, b)} = a + b^\top x.$$ 

- Solve explicitly for conditional class probabilities (using $p(Y = +1|X = x; a, b) + p(Y = -1|X = x; a, b) = 1$):

$$p(Y = +1|X = x; a, b) = \frac{1}{1 + \exp(-(a + b^\top x))} =: s(a + b^\top x)$$

$$p(Y = -1|X = x; a, b) = \frac{1}{1 + \exp((a + b^\top x))} = s(-a - b^\top x)$$

where $s(z) = 1/(1 + \exp(-z))$ is the **logistic function**.
Fitting the parameters of the hyperplane

How to learn $a$ and $b$ given a training data set $(x_i, y_i)_{i=1}^n$?

- Consider maximizing the **conditional log likelihood** for $Y = \{+1, -1\}$:

$$p(Y = y_i | X = x_i; a, b) = p(y_i | x_i) = \begin{cases} s(a + b^\top x_i) & \text{if } Y = +1 \\ 1 - s(a + b^\top x_i) & \text{if } Y = -1 \end{cases}$$

- Noting that $1 - s(z) = s(-z)$, we can write the log-likelihood using the compact expression:

$$\log p(y_i | x_i) = \log s(y_i(a + b^\top x_i)).$$

- And the log-likelihood over the whole i.i.d. data set is:

$$\ell(a, b) = \sum_{i=1}^n \log p(y_i | x_i) = \sum_{i=1}^n \log s(y_i(a + b^\top x_i)).$$
Fitting the parameters of the hyperplane

How to learn \( a \) and \( b \) given a training data set \((x_i, y_i)_{i=1}^n\) ?

- Consider maximizing the **conditional log likelihood**:

  \[
  \ell(a, b) = \sum_{i=1}^{n} \log p(y_i | x_i) = \sum_{i=1}^{n} \log s(y_i(a + b^T x_i)).
  \]

- Equivalent to minimizing the empirical risk associated with the **log loss**:

  \[
  \hat{R}_{\log}(f_{a,b}) = \frac{1}{n} \sum_{i=1}^{n} - \log s(y_i(a + b^T x_i)) = \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-y_i(a + b^T x_i))).
  \]
Could we use the 0-1 loss?

- With the 0-1 loss, the risk becomes:

\[
\hat{R}(f_{a,b}) = \frac{1}{n} \sum_{i=1}^{n} \text{step}(-y_i(a + b^\top x_i))
\]

- But what is the gradient? ...