Lecture 16: PCA, LDA, and QDA

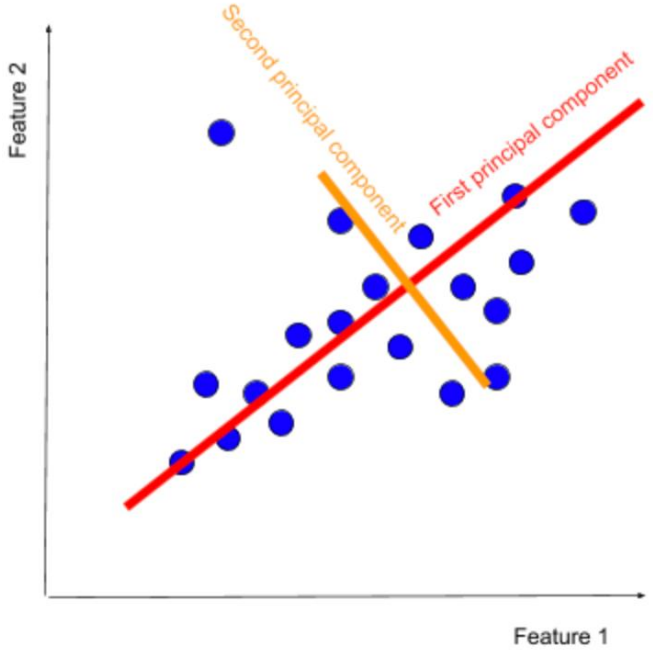
Instructor: Sergei V. Kalinin

What does it take to make ML useful?

Principal Component Analysis

PCA: orthogonal transformation converting possibly correlated variables into linearly uncorrelated *principal components*

- PCA transforms the data such that the greatest variance by any projection lies on the first coordinate
- The first PC retains the greatest amount of variation in the sample. The k^{th} PC retains the k^{th} greatest fraction of the variation in the sample.
- The k^{th} largest eigenvalue of the correlation matrix C is the variance in the sample along the k^{th} PC
- Reveals internal structure that best explains variance in the dataset



The least-squares view: PCs are a series of linear least squares fits to a sample, each orthogonal to all previous ones

https://medium.com/@kayalgen/principal-component-analysis-pca-for-machine-learning-89db6cb63e4

PCA in scikit-learn

sklearn.decomposition.PCA

class sklearn.decomposition.**PCA**(*n_components=None, *, copy=True, whiten=False, svd_solver='auto', tol=0.0, iterated_power='auto', n_oversamples=10, power_iteration_normalizer='auto', random_state=None*) [source]

Methods

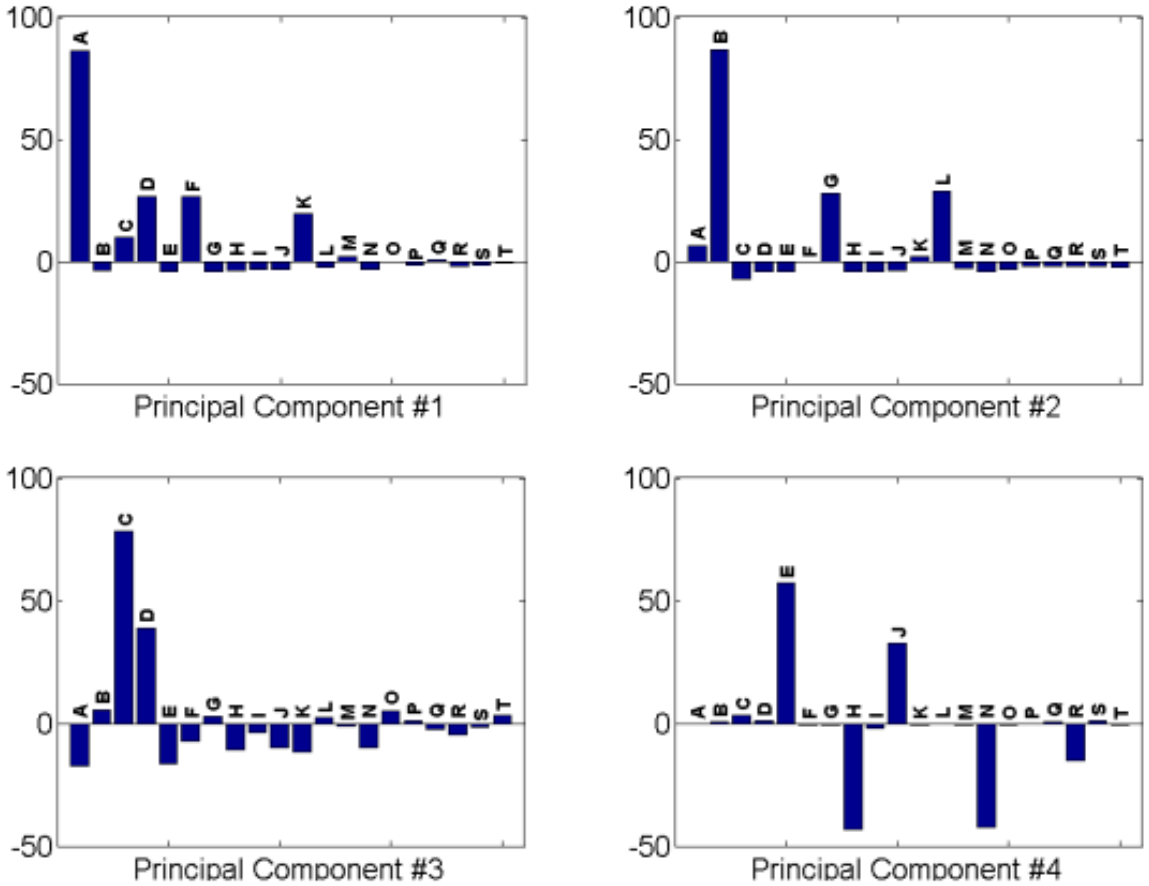
fit(X[, y])	Fit the model with X.
fit_transform(X[, y])	Fit the model with X and apply the dimensionality reduction on X.
get_covariance()	Compute data covariance with the generative model.
get_feature_names_out([input_features])	Get output feature names for transformation.
get_metadata_routing()	Get metadata routing of this object.
get_params([deep])	Get parameters for this estimator.
get_precision()	Compute data precision matrix with the generative model.
inverse_transform(X)	Transform data back to its original space.
score(X[, y])	Return the average log-likelihood of all samples.
score_samples(X)	Return the log-likelihood of each sample.
set_output(*[, transform])	Set output container.
set_params(**params)	Set the parameters of this estimator.
transform(X)	Apply dimensionality reduction to X.

- In scikit-learn, PCA is implemented as **probabilistic model**: can estimate the likelihood of data based on the amount of variance it explains.
- **Incremental PCA:** add partial_fit method
- **SparsePCA:** introduce sparsity in decomposition

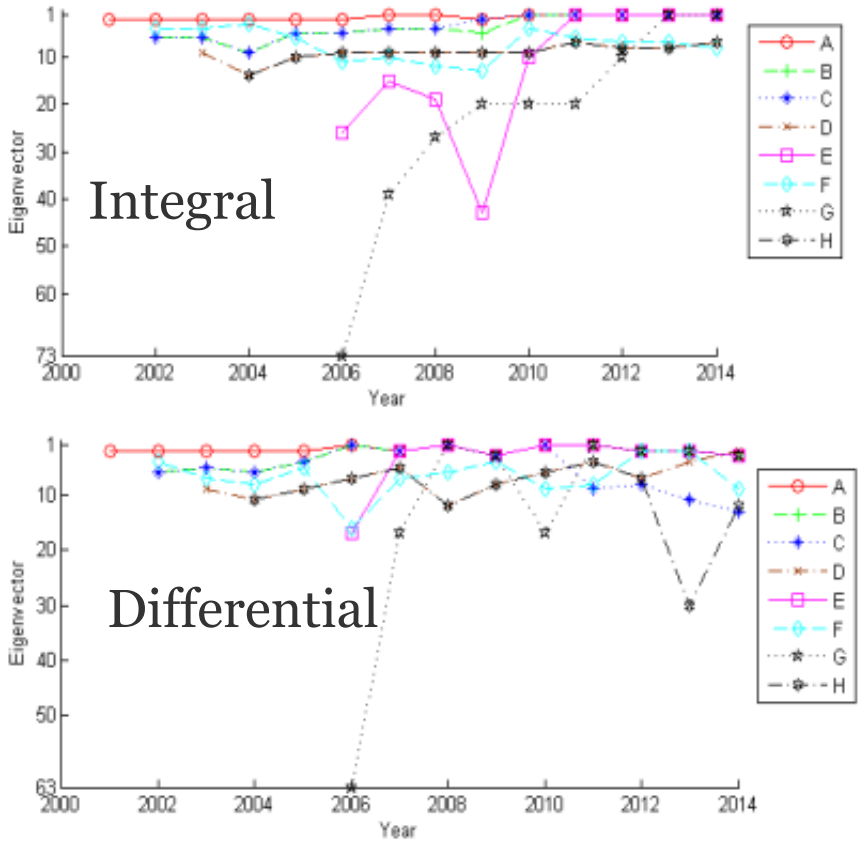
$$(U^*, V^*) = \underset{U,V}{\arg\min} \frac{1}{2}||X - UV||_{\text{Fro}}^2 + \alpha||V||_{1,1}$$
$$\text{subject to } ||U_k||_2 <= 1 \text{ for all } 0 \leq k < n_{components}$$

- … and there is always more (dictionary methods, etc)

PCA can be applied to everything



Publication activity



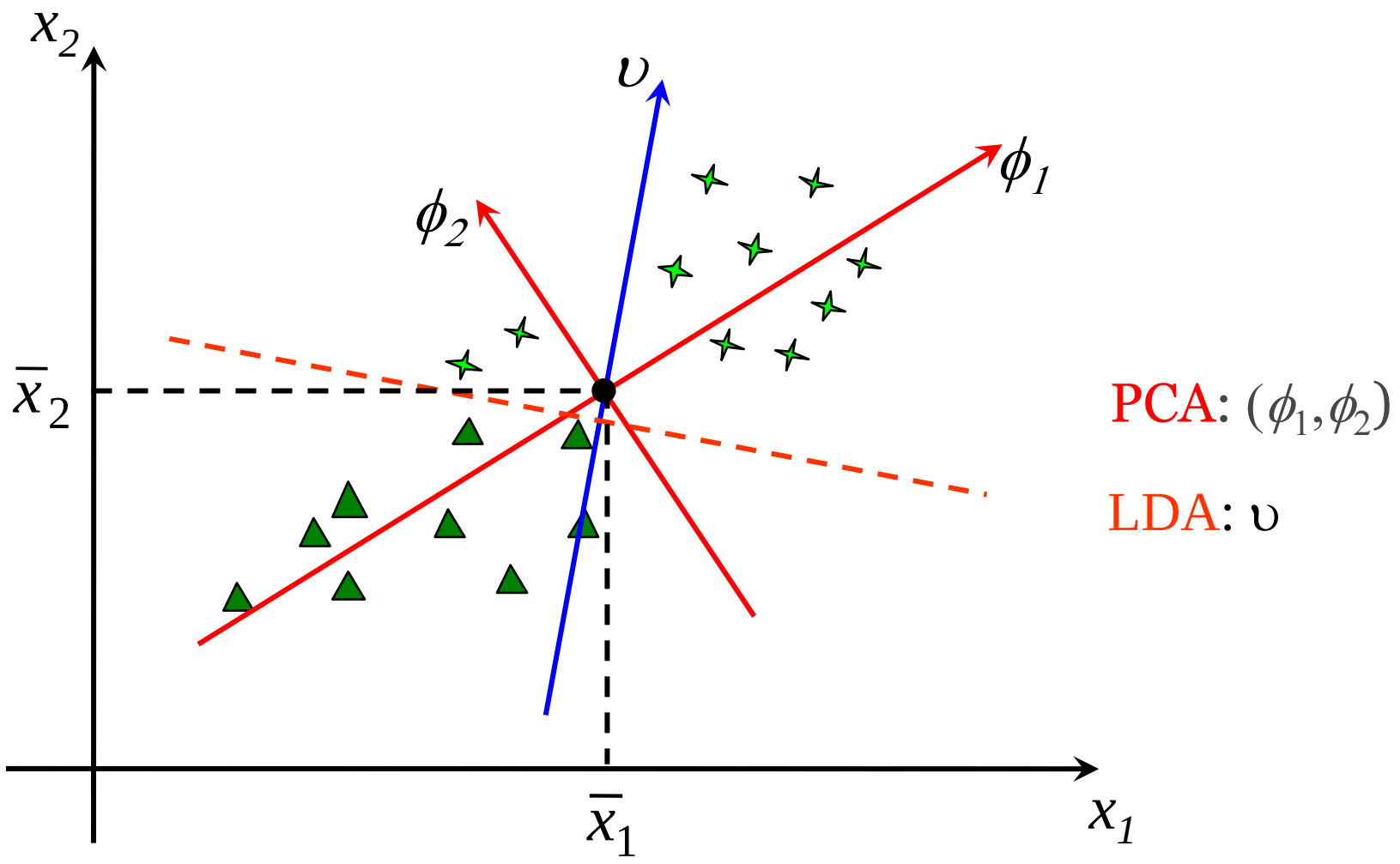
Publication analysis: The dimensionality of space, N, is defined by the total number of scientists. Each publication in this case then defines a single point in this space. For example, for the 28-dimensional space of authors of {A,B,C, … , Z} the publication authored by A and C will be represented as {1,0,1, …, 0}, and by A,B, and Z as {1,1,0, …, 0,1}, where all missing elements are zeroes.

https://arxiv.org/abs/1502.03439

Linear Discriminant Analysis

- Linear Discriminant Analysis, or simply LDA, is a feature extraction technique that has been used successfully in many statistical pattern recognition problems.
- LDA is often called Fisher Discriminant Analysis (FDA).
- The primary purpose of LDA is to separate samples of distinct groups by transforming then to a space which maximises their between-class separability while minimising their within-class variability.
- It assumes implicitly that the true covariance matrices of each class are equal because the same within-class scatter matrix is used for all the classes considered.
- If we remove this assumption, we get Quadratic Discriminant Analysis

Geometric Idea of LDA



From Intelligent Data Analysis and Probabilistic Inference by Longin Jan Latecki Temple University

LDA Steps

1. Compute the d-dimensional mean vectors.
2. Compute the scatter matrices
3. Compute the eigenvectors and corresponding eigenvalues for the scatter matrices.
4. Sort the eigenvalues and choose those with the largest eigenvalues to form a $d \times k$ dimensional matrix
5. Transform the samples onto the new subspace.

LDA Method

- Let the between-class scatter matrix S_b be defined as

$$S_b = \sum_{i=1}^{g} N_i (\bar{x}_i - \bar{x})(\bar{x}_i - \bar{x})^T$$

- and the within-class scatter matrix S_w be defined as

$$S_w = \sum_{i=1}^{g} (N_i - 1) S_i = \sum_{i=1}^{g} \sum_{j=1}^{N_i} (x_{i,j} - \bar{x}_i)(x_{i,j} - \bar{x}_i)^T$$

- where $x_{i,j}$ is an n-dimensional data point j from class p_i, N_i is the number of training examples from class p_i, and g is the total number of classes or groups.

From Intelligent Data Analysis and Probabilistic Inference by Longin Jan Latecki Temple University

LDA Method

- The sample mean, sample covariance, and grand mean vector are given respectively by:

$$\bar{x}_i = \frac{1}{N_i}\sum_{j=1}^{N_i} x_{i,j}$$

$$S_i = \frac{1}{(N_i - 1)}\sum_{j=1}^{N_i}(x_{i,j} - \bar{x}_i)(x_{i,j} - \bar{x}_i)^T$$

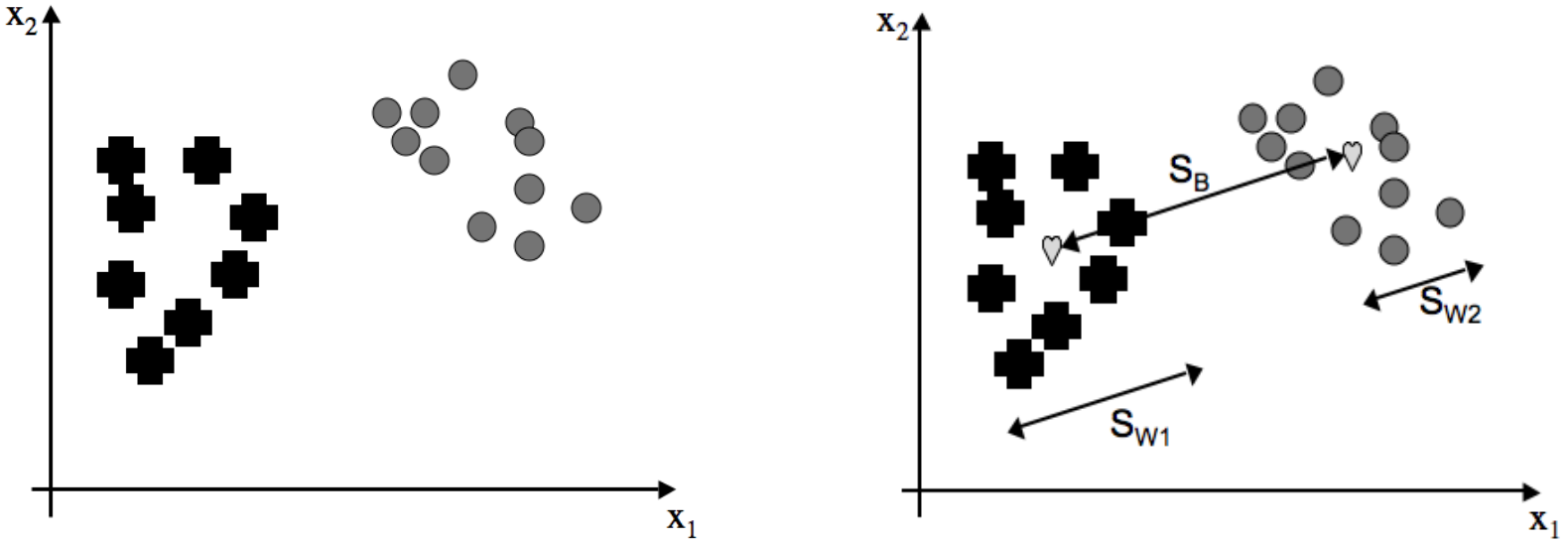
$$\bar{x} = \frac{1}{N}\sum_{i=1}^{g} N_i \bar{x}_i = \frac{1}{N}\sum_{i=1}^{g}\sum_{j=1}^{N_i} x_{i,j} \qquad N = N_1 + N_2 + \ldots + N_g$$

From Intelligent Data Analysis and Probabilistic Inference by Longin Jan Latecki Temple University

LDA Method

The objective of LDA is to find a projection matrix P_{lda} that maximises the ratio of the determinant of S_b to the determinant of S_w (Fisher's criterion) as:

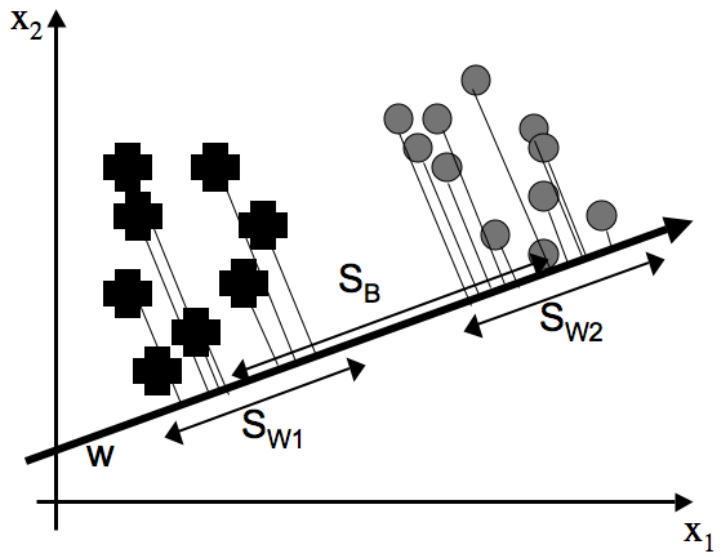
$$P_{lda} = \arg\max_{P} \frac{\left| P^T S_b P \right|}{\left| P^T S_w P \right|}$$



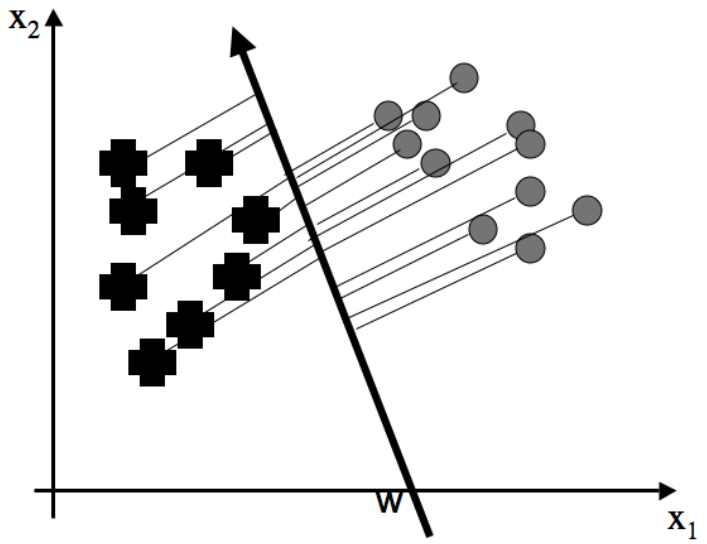
From Intelligent Data Analysis and Probabilistic Inference by Longin Jan Latecki Temple University

LDA Method

- So Fisher's criterion tries to find the projection that:
  - Maximises the variance of the class means
  - Minimises the variance of the individual classes



Good projection



Bad projection

From Intelligent Data Analysis and Probabilistic Inference by Longin Jan Latecki Temple University

Classification using LDA

- The LDA is an axis projection.
- Once the projection is found all the data points can be transformed to the new axis system along with the class means and covariances.
- Allocation of a new point to a class can be done using a distance measure such as the Mahalanobis distance.

Given a vector $\mathbf{x}$, a mean vector μ, and a covariance matrix $\mathbf{\Sigma}$, the Mahalanobis Distance D_M is defined as:

$$D_M(\mathbf{x}, \mu) = \sqrt{(\mathbf{x} - \mu)^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \mu)}$$

where:

- $\mathbf{x}$ is the vector for which the Mahalanobis Distance is being computed.
- μ is the mean vector of the distribution.
- $\mathbf{\Sigma}$ is the covariance matrix of the distribution.
- ^{T} denotes a matrix transpose.
- $\mathbf{\Sigma}^{-1}$ is the inverse of the covariance matrix.

From Intelligent Data Analysis and Probabilistic Inference by Longin Jan Latecki Temple University

LDA vs. PCA

- LDA is supervised ML method, PCA is unsupervised method
- LDA seeks directions that are efficient for *discriminating* data whereas PCA seeks directions that are efficient for *representing* data.
- The directions that are discarded by PCA might be exactly the directions that are necessary for distinguishing between groups.

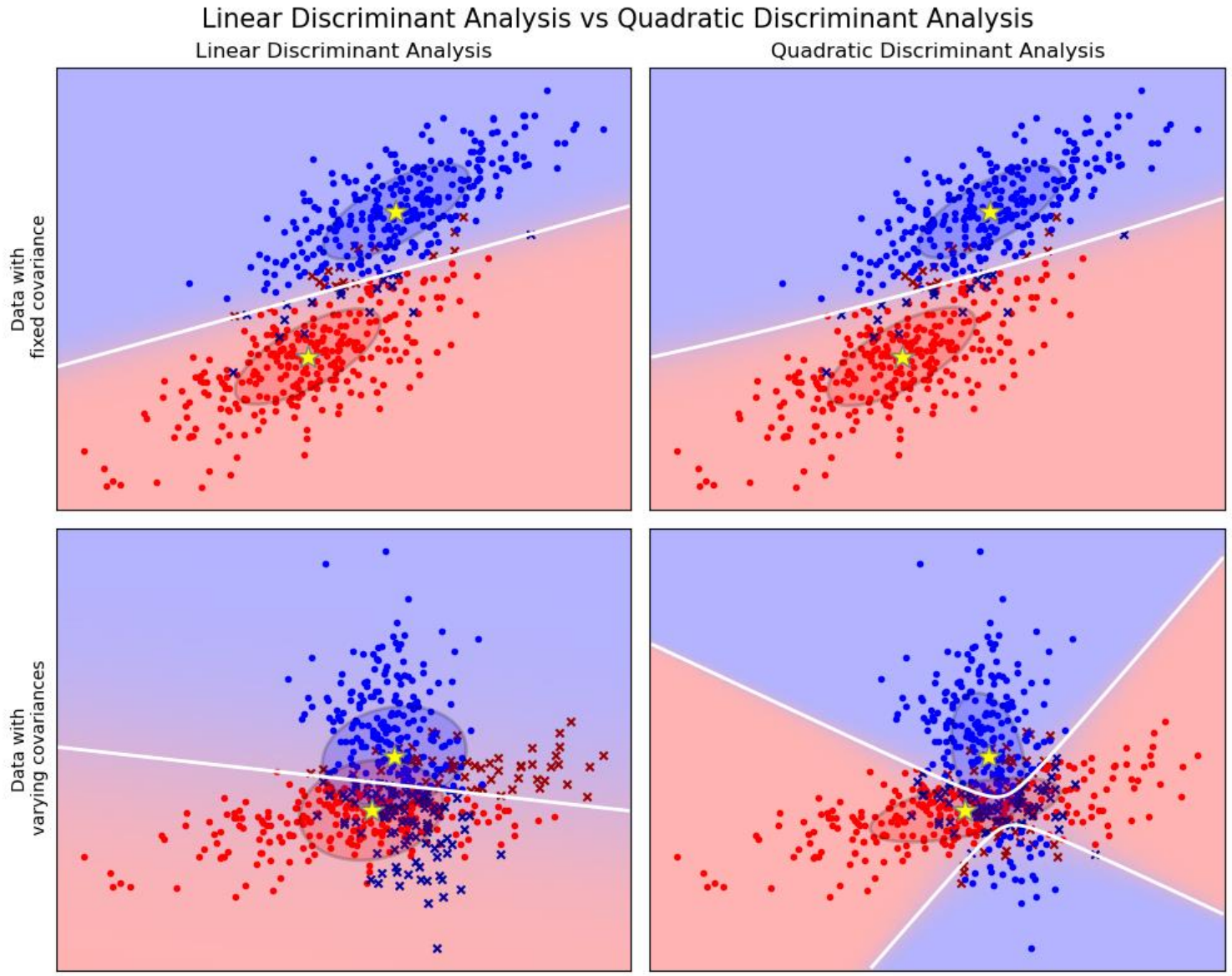
From Intelligent Data Analysis and Probabilistic Inference by Longin Jan Latecki Temple University

sklearn.discriminant_analysis.LinearDiscriminantAnalysis

class sklearn.discriminant_analysis.**LinearDiscriminantAnalysis**(*solver='svd', shrinkage=None, priors=None, n_components=None, store_covariance=False, tol=0.0001, covariance_estimator=None*) [source]

decision_function(X)	Apply decision function to an array of samples.
fit(X, y)	Fit the Linear Discriminant Analysis model.
fit_transform(X[, y])	Fit to data, then transform it.
get_feature_names_out([input_features])	Get output feature names for transformation.
get_metadata_routing()	Get metadata routing of this object.
get_params([deep])	Get parameters for this estimator.
predict(X)	Predict class labels for samples in X.
predict_log_proba(X)	Estimate log probability.
predict_proba(X)	Estimate probability.
score(X, y[, sample_weight])	Return the mean accuracy on the given test data and labels.
set_output(*[, transform])	Set output container.
set_params(**params)	Set the parameters of this estimator.
set_score_request(*[, sample_weight])	Request metadata passed to the score method.
transform(X)	Project data to maximize class separation.

Quadratic Discriminant Analysis



https://scikit-learn.org/stable/modules/lda_qda.html#lda-qda

LDA vs. QDA

Bayesian classification framework:

$$P(y=k|x) = \frac{P(x|y=k)P(y=k)}{P(x)} = \frac{P(x|y=k)P(y=k)}{\sum_l P(x|y=l)\cdot P(y=l)}$$

QDA approximation:

$$P(x|y=k) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_k)^t \Sigma_k^{-1}(x-\mu_k)\right)$$

LDA is a special case of QDA, where the Gaussians for each class are assumed to share the same covariance matrix: $\Sigma_k = \Sigma$. If so

$$\log P(y=k|x) = \omega_k^t x + \omega_{k0} + Cst.$$

where $\omega_k = \Sigma^{-1}\mu_k$ and $\omega_{k0} = -\frac{1}{2}\mu_k^t \Sigma^{-1}\mu_k + \log P(y=k)$. These quantities correspond to the `coef_` and `intercept_` attributes, respectively.

Note: Relation with Gaussian Naive Bayes

If in the QDA model one assumes that the covariance matrices are diagonal, then the inputs are assumed to be conditionally independent in each class, and the resulting classifier is equivalent to the Gaussian Naive Bayes classifier `naive_bayes.GaussianNB`.

https://scikit-learn.org/stable/modules/lda_qda.html#lda-qda