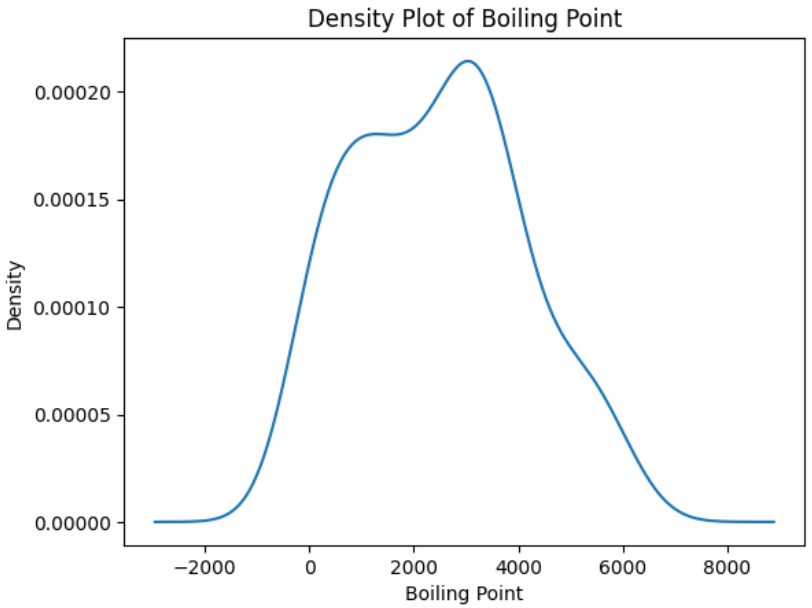
Lecture 17: More Linear

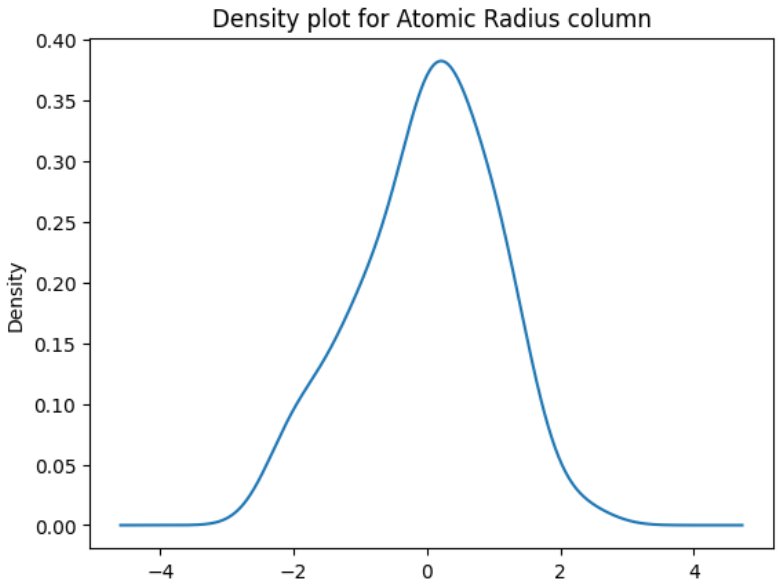
Instructor: Sergei V. Kalinin

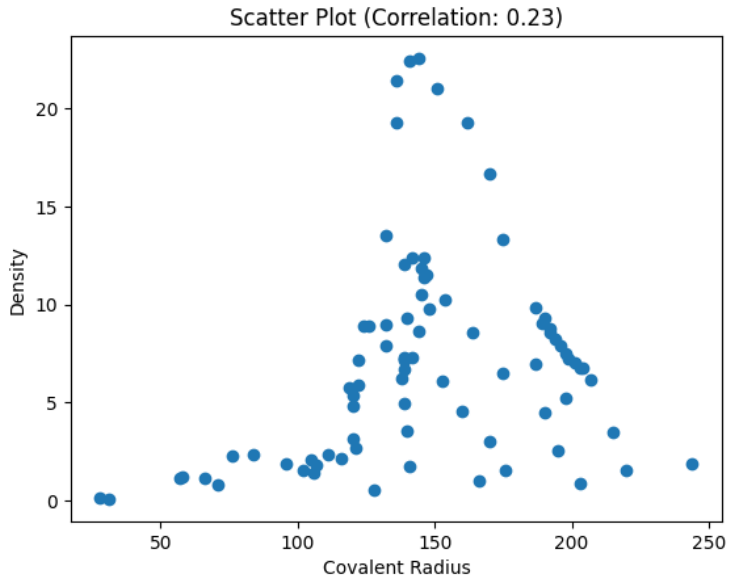
Homeworks 2 and 3:

- **Homework 2:** Some examples of ML for materials are very interesting
- Please remember to put everything in Colabs
- Save Colabs with all outputs
- And Colab should be run from beginning to the end

Subtle moments:

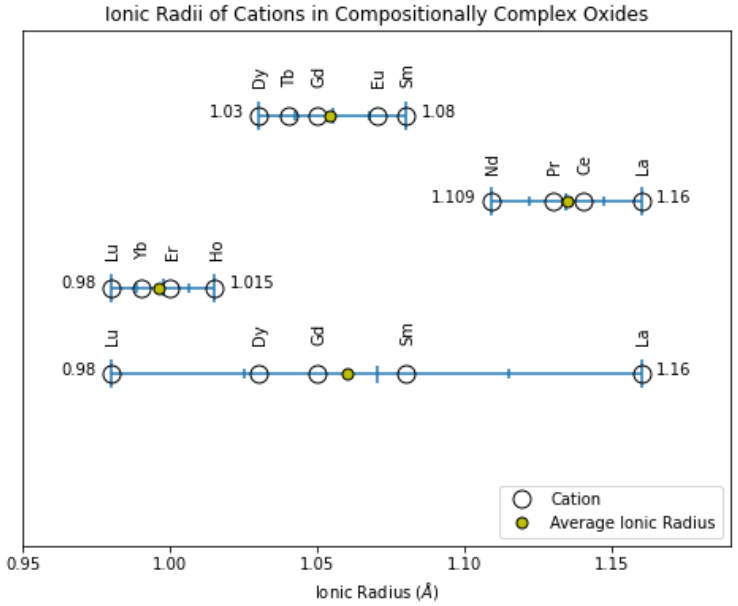


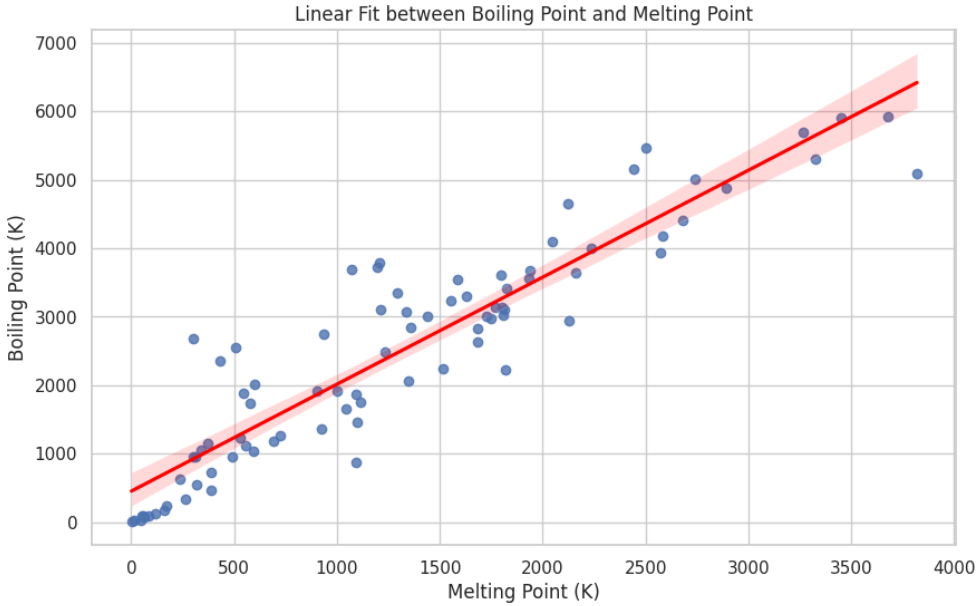


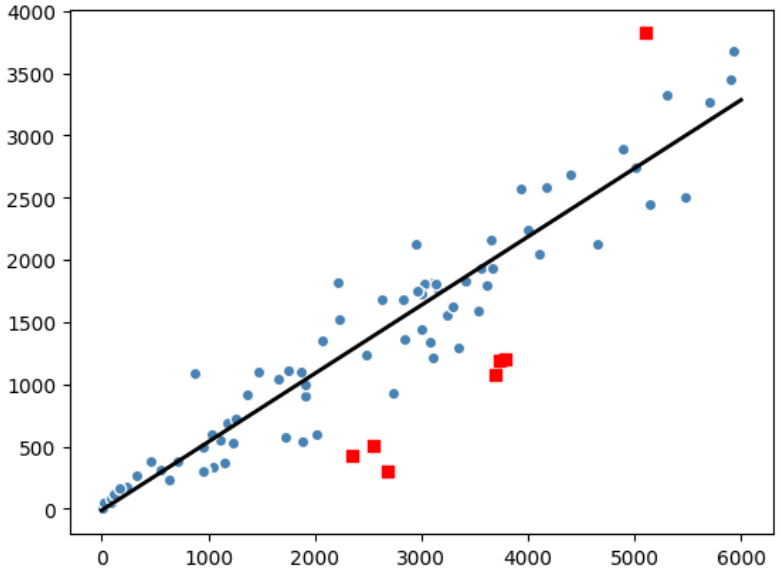


- When using ML, always check if the answers are physical
- Also, please answer both the ML question and physics question

Impressive answers:

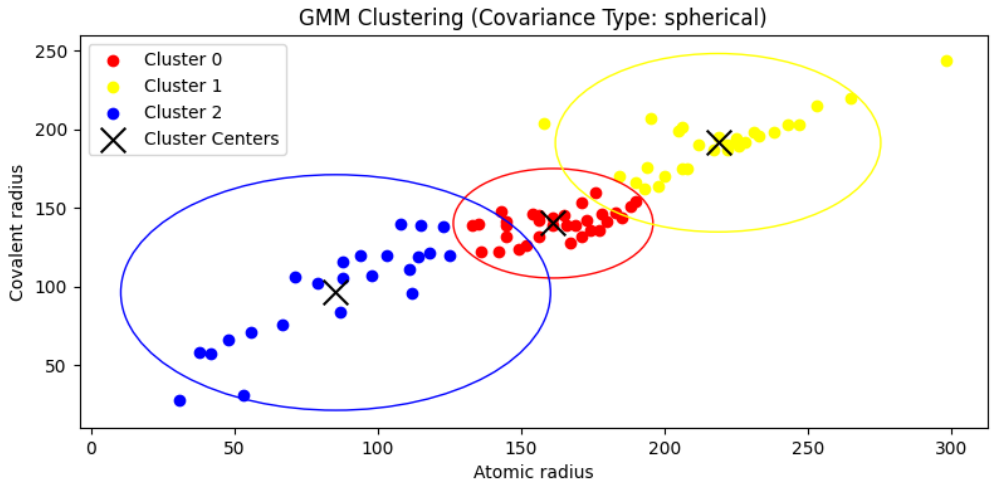


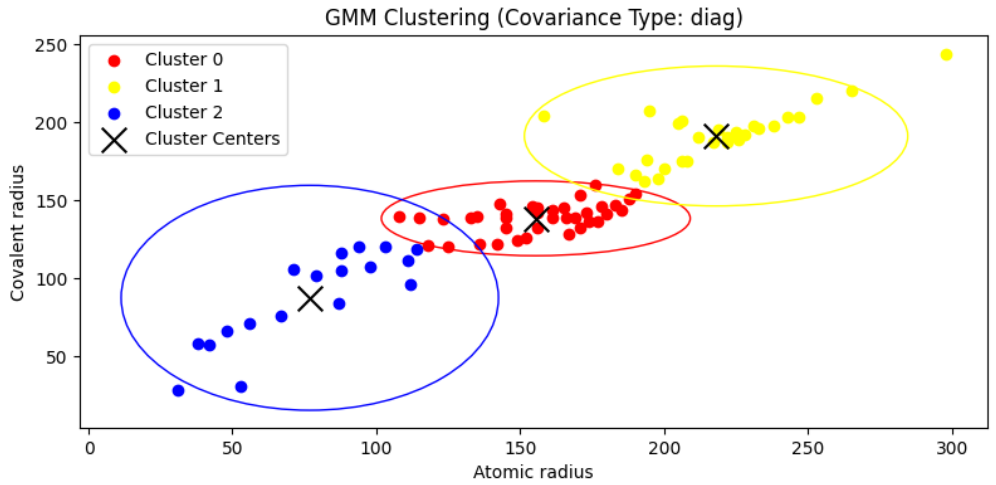


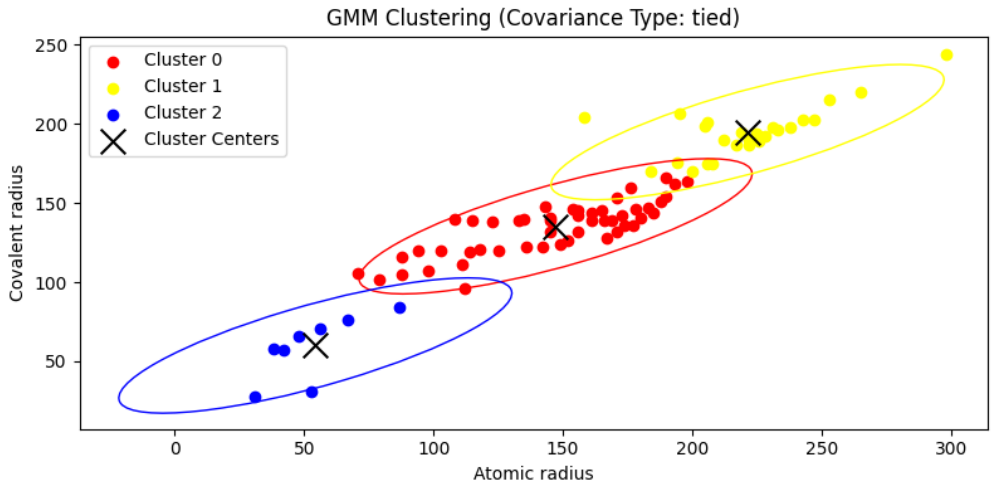


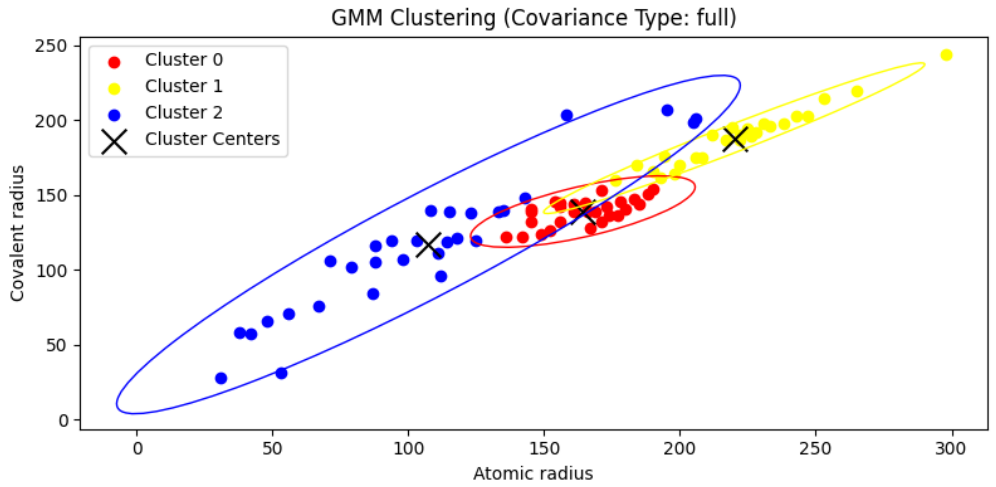
… and many more!

Impressive answers:

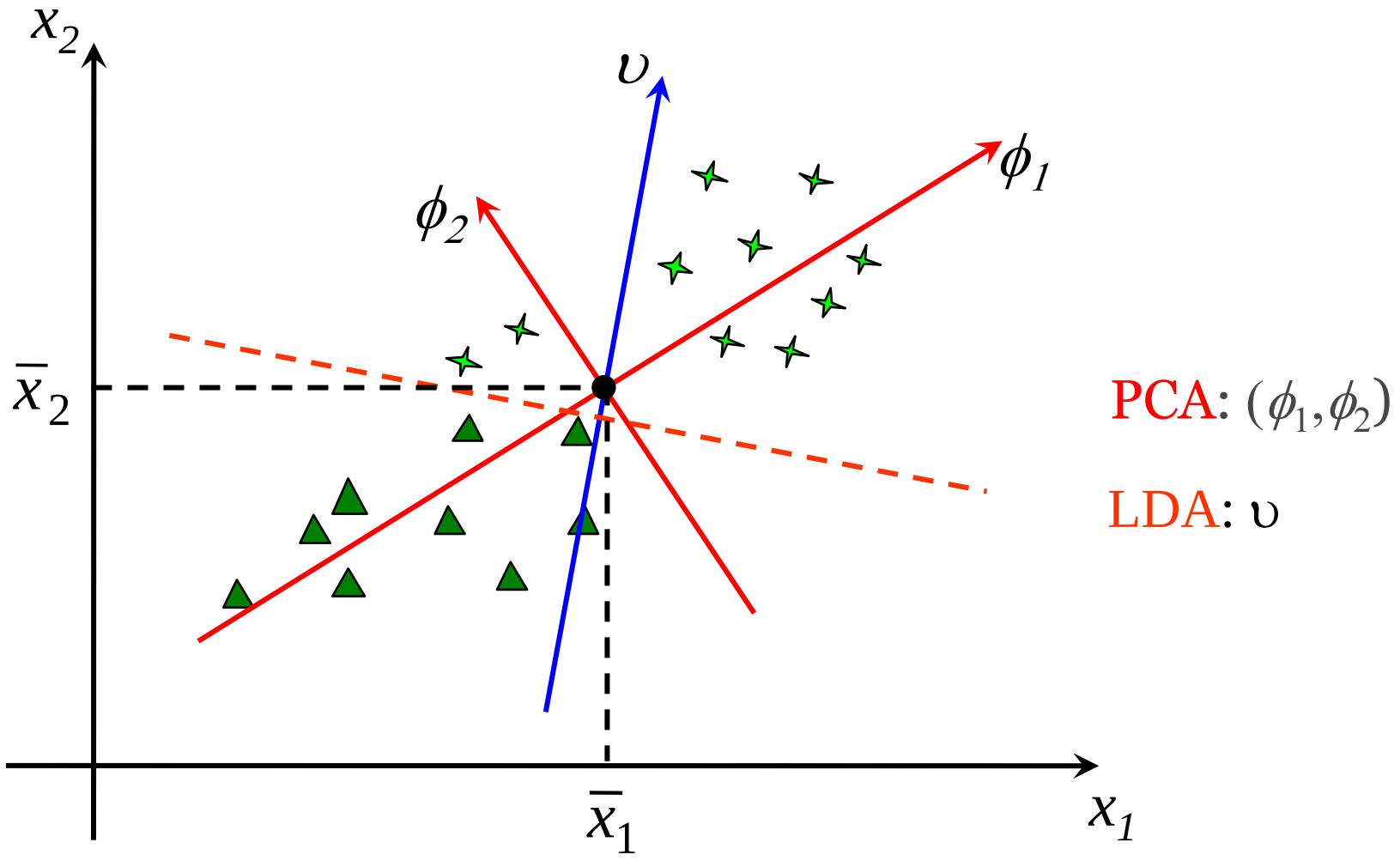








PCA and LDA



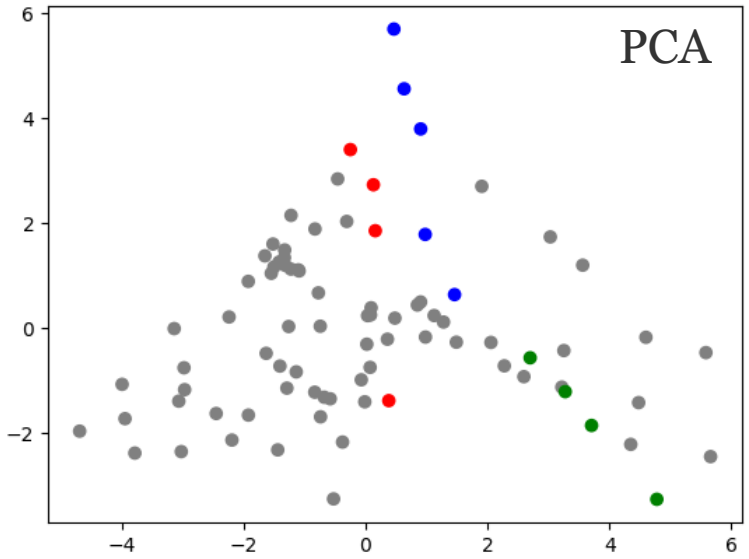
From Intelligent Data Analysis and Probabilistic Inference by Longin Jan Latecki Temple University

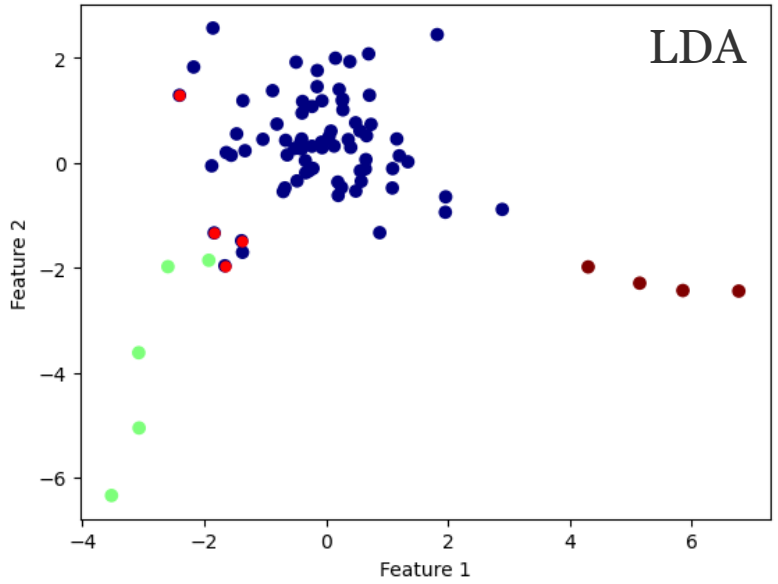
PCA vs. LDA for elements

Alkali

Alkali earth

Halogens

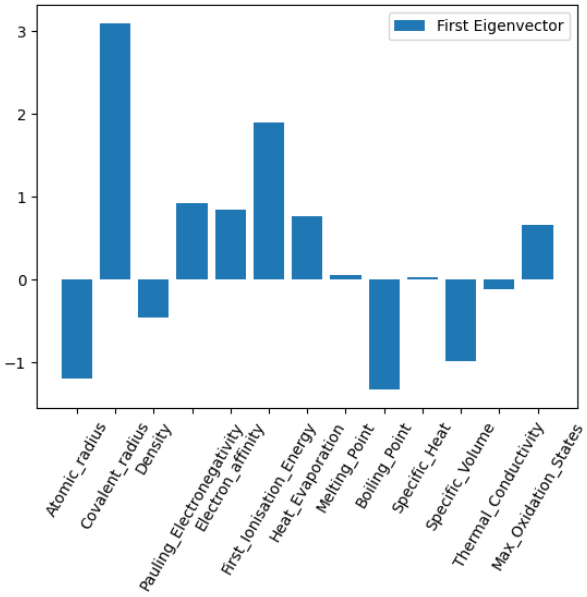




Alkali

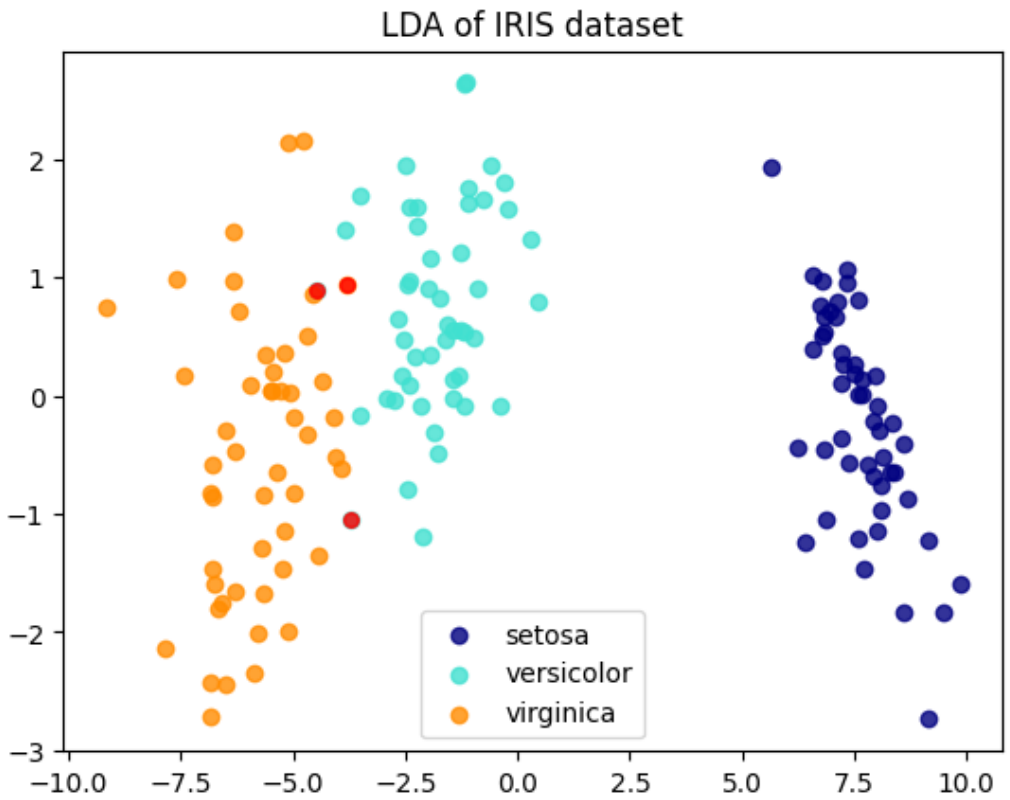
Alkali earth

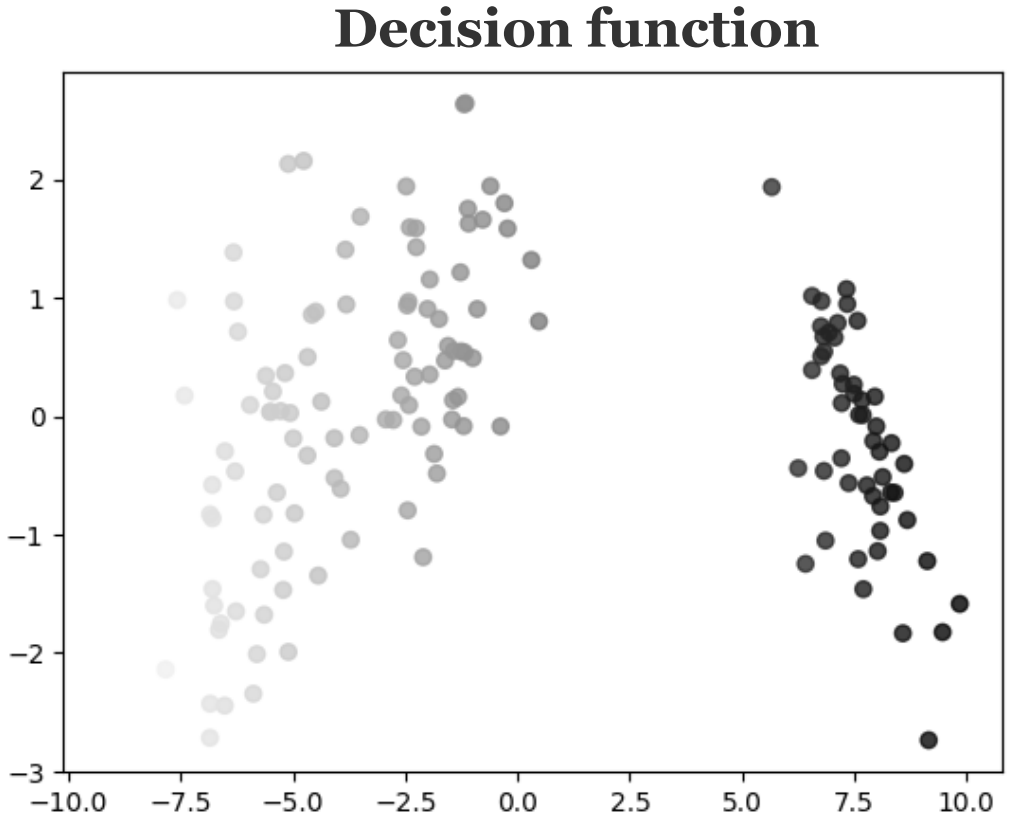
Halogens



- Our element space is described by 13 descriptors
- In PCA, we found 2 linear combinations of these descriptors that describe this data set best.
- Alkali, alkali-earth, and halogens are close to each other in PCA space
- LDA finds best representation to separate alkali and halogens from everything else
- In LDA representation, alkali earth are close to alkali

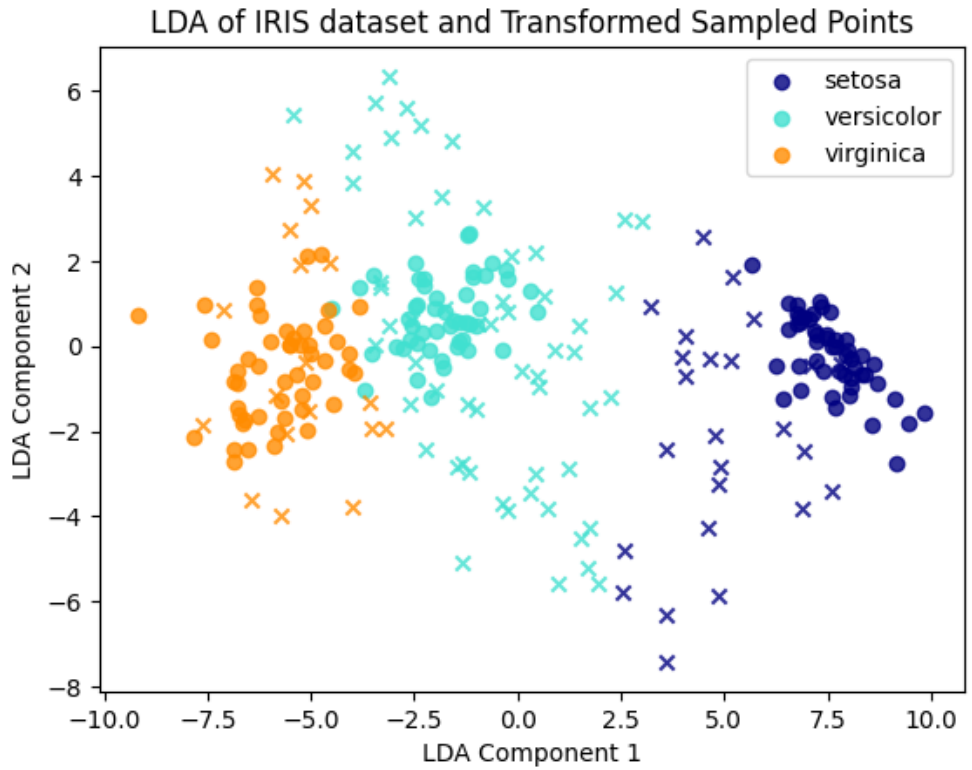
What else can LDA give us?

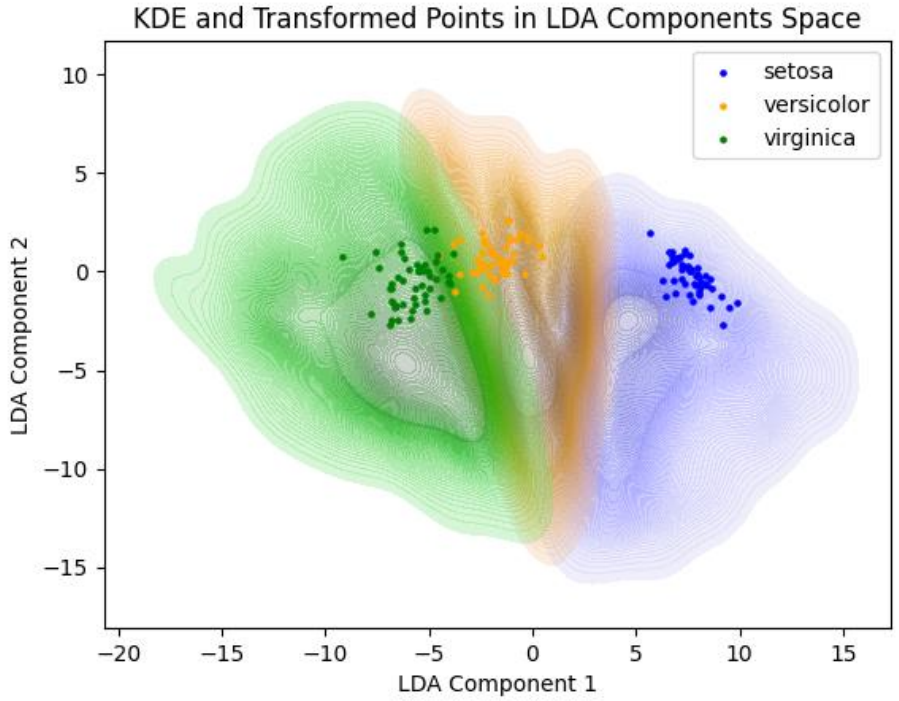




- Decision function allows us to quantify how likely is the feature to belong to certain class

Visualizing the decision surfaces





- Generate multiple points uniformly distributed in the original high-dimensional space
- Perform the LDA transform
- Calculate the KDE

Visualizing the decision surfaces

10_PCA_LDA_Elements.ipynb

And one transform to rule them all…

https://en.wikipedia.org/wiki/One_Ring

Canonical Correlation Analysis

- Simple correlation: $y_1 \sim x_1$
- Multiple correlation: $y_1 \sim x_1 \quad x_2 \quad x_3$
- Canonical correlation: $y_1 \quad y_2 \quad y_3 \sim x_1 \quad x_2 \quad x_3$

- Start with multiple y and x variables: $y_1 \quad y_2 \quad y_3 \sim x_1 \quad x_2 \quad x_3$
- Construct a "canonical variate" as the combination of y variables:

$$CV_{y1} = b_1 y_1 + b_2 y_2 + b_3 y_3$$

- Construct a "canonical variate" as the combination of x variables

$$CV_{x1} = a_1 x_1 + a_2 x_2 + a_3 x_3$$

- The canonical correlation is the correlation of the variates $R_c = r_{cvy1,\ cvx1}$
- Find a's and b's to maximize R_c
- Iterate for second pair

Steps of CCA

1. **Start with two datasets X and Y:** Standardize if necessary (subtract the mean and divide by the standard deviation for each variable).
2. Find the **First Pair of Canonical Variables** by maximizing the correlation between linear combinations of **X** and **Y**. Use optimization algorithm (e.g., an eigenvalue or SVD) to find the weight vectors **a** and **b** for **X** and **Y** respectively, such that the correlation between **X a** and **Y b** is maximized.
3. **Orthogonalization:** Find the residuals by removing the projections on the first pair of canonical variables from **X** and **Y** respectively. This can be done by regressing each variable on the first canonical variable and taking the residuals.
4. **Next Pair of Canonical Variables:** Repeat the CCA process with the residual data to find the next pair of canonical variables. Again, find weight vectors **a2** and **b2** for the residuals of X and Y respectively, that maximize the correlation between the new linear combinations.
5. **Continue the Process** iteratively, orthogonalizing the datasets at each step and performing CCA on the residuals, until the desired number of canonical variable pairs is obtained or until the correlations for new pairs are near zero.
6. **Result**: Obtain multiple pairs of canonical variables, each pair being uncorrelated with the others.

Canonical Correlation Analysis

Canonical correlation is used (instead of comparisons among selected simple and/or multiple correlations) when:

- We are interested in which combination(s) of feature variables are related to which combination(s) of target variables
- Whether the set of variables have a concentrated or a diffuse correlational structure

For example: we have two sets of predictor variables and want to show that one set is most useful for understanding the one criterion variable and the other set is most useful for understanding a different set of criterion variables

CCA in scikit-learn

sklearn.cross_decomposition.CCA

class sklearn.cross_decomposition.**CCA**(*n_components=2, *, scale=True, max_iter=500, tol=1e-06, copy=True*) [source]

fit(X, Y)	Fit model to data.
fit_transform(X[, y])	Learn and apply the dimension reduction on the train data.
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.
inverse_transform(X[, Y])	Transform data back to its original space.
predict(X[, copy])	Predict targets of given samples.
score(X, y[, sample_weight])	Return the coefficient of determination of the prediction.
set_output(*[, transform])	Set output container.
set_params(**params)	Set the parameters of this estimator.
set_predict_request(*[, copy])	Request metadata passed to the `predict` method.
set_score_request(*[, sample_weight])	Request metadata passed to the `score` method.
set_transform_request(*[, copy])	Request metadata passed to the `transform` method.
transform(X[, Y, copy])	Apply the dimension reduction.

CCA Outputs

- **x_weights_:** Contains the canonical weights for the first dataset (X). Used to compute the canonical variables for X.
- **y_weights_:** Contains the canonical weights for the second dataset (Y). Used to compute the canonical variables for Y.
- **x_scores_:** Contains the scores of X, i.e., the canonical variables derived from X. Calculated by multiplying X with x_weights_.
- **y_scores_:** Contains the scores of Y, i.e., the canonical variables derived from Y. Calculated by multiplying Y with y_weights_.
- **x_loadings_:** Contains the canonical loadings for X. Represents the correlation between X and the canonical variables of X.
- **y_loadings_:** Contains the canonical loadings for Y. Represents the correlation between Y and the canonical variables of Y.
- **coef_:** Contains the coefficients of the linear model used to transform X to Y. Represents the relationship between the canonical variables of X and Y.

Why these methods are useful?

Pro:

1.Computational Efficiency
2. Interpretability
3. Less Data Requirement
4. Applicability to Linear Problems
5. Ease of Implementation and Use
6. Feature Reduction and Visualization
7. Foundation for More Complex Models

Con:

1. Assumption of Linearity
2. Assumption of Gaussian distributions
3. Limited Complexity
4. Independence Assumption in PCA and CCA
5. Vulnerability to Outliers
6. Feature Importance Ambiguity

Linear methods in Deep Learning Era

1. Preprocessing: To reduce the dimensionality of data or extract more informative features before feeding them into a neural network.

2. Visualization: To visualize high-dimensional embeddings learned by neural networks.

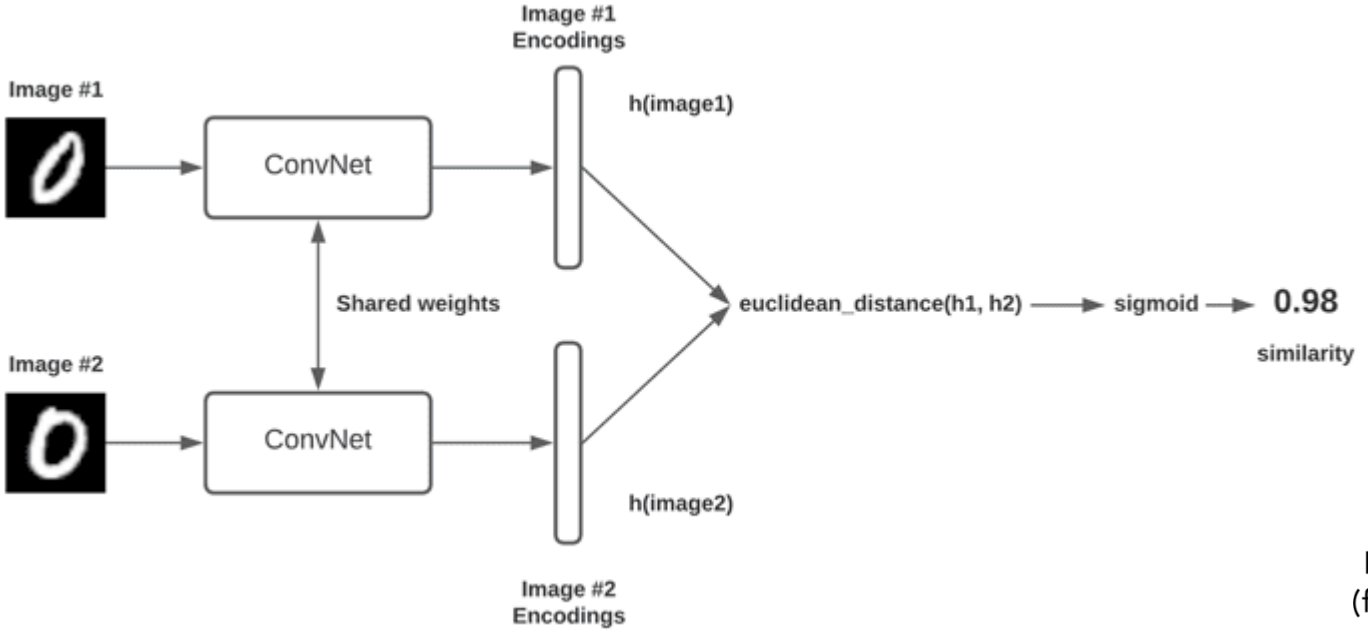
3. Post-Processing of Embeddings: To further process or analyze embeddings learned by a neural network for various tasks. Specifically, **CCA** can be used to measure the similarity of learned representations from different neural networks or layers, often used in analyzing and comparing embeddings.

4. Improving Model Robustness and Generalization: To enhance the robustness and generalization capabilities of deep learning models: feature extraction or transformation, potentially enhancing the robustness and generalization of neural network models.

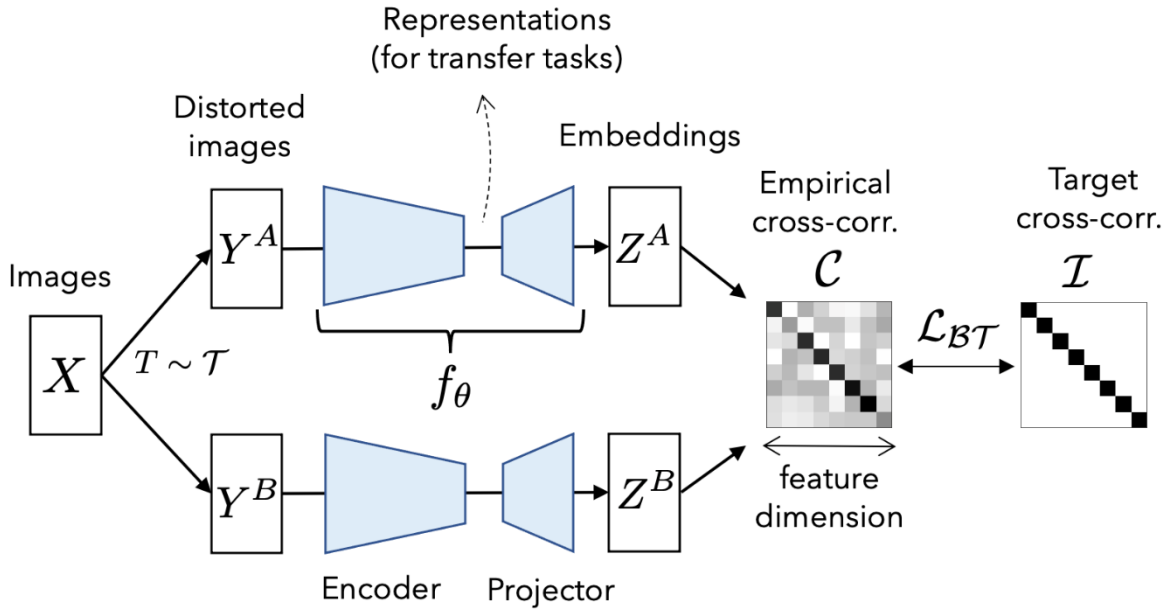
5. Facilitating Transfer Learning: To adapt learned representations for transfer learning applications, e.g. transform embeddings for alignment, compatibility, or adaptation

6. Interpreting Neural Network Decisions: To interpret and understand the decisions or representations learned by neural networks, e.g. analyze and interpret the feature representations or activations within a neural network, contributing to model interpretability and transparency.

Siamese networks



Barlow twins



https://pyimagesearch.com/2020/11/30/siamese-networks-with-keras-tensorflow-and-deep-learning/

https://github.com/facebookresearch/barlowtwins

11_PCA_GenerativeModel_Images.ipynb