2.2 - Feature Engineering and Dimensionality Reduction

Giovanni Della Lunga
giovanni.dellalunga@unibo.it

Introduction to Machine Learning for Finance

Bologna - February-March, 2025

Subsection 1

Regularization

Regularization

- Linear regression can over-fit, particularly when there are a large number of correlated features.
- Results for validation set may not then be as good as for training set
- Regularization is a way of avoiding overfitting and reducing the number of features. Alternatives:
  - **Ridge Regression**
  - **Lasso Regression**
  - **Elastic net**
- We must first scale feature values

Transforming Polynomial Regression into Linear Regression

Regularization

- Polynomial regression allows modeling nonlinear relationships.
- Despite its name, it can be rewritten as a linear regression.
- This is done by redefining input features.

Polynomial Regression Model

General form of polynomial regression:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_d x^d + \epsilon \tag{1}$$

- y: dependent variable (target)
- x: independent variable (feature)
- $\beta_0, \beta_1, \ldots, \beta_d$: regression coefficients
- ϵ: error term

Transforming Polynomial Regression into Linear Regression

Regularization

Define new feature variables:

$$X_1 = x, \quad X_2 = x^2, \quad X_3 = x^3, \quad \ldots, \quad X_d = x^d$$

Rewritten as:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_d X_d + \epsilon \tag{2}$$

- Now, the model is linear in parameters β.
- We can apply standard linear regression techniques.

Transforming Polynomial Regression into Linear Regression

Regularization

Rewriting polynomial regression in matrix form:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{3}$$

Where:

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^d \\ 1 & x_2 & x_2^2 & \dots & x_2^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^d \end{bmatrix}$$

Transforming Polynomial Regression into Linear Regression

Regularization

Using Scikit-Learn to transform features and fit a linear model:

```
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

# Sample data
X = np.array([[1], [2], [3], [4], [5]])
y = np.array([2, 5, 10, 17, 26])  # Quadratic relationship

# Transform X to polynomial features (degree 2)
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)

# Fit linear regression on transformed features
model = LinearRegression()
model.fit(X_poly, y)
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
```

Transforming Polynomial Regression into Linear Regression

Regularization

- Polynomial regression is linear in parameters, despite nonlinear terms.
- Transform features to apply linear regression techniques.
- Regularization (Ridge, Lasso) can help prevent overfitting.

Ridge Regression

- Ridge regression is a regularization technique where we change the function that is to be minimize;
- Reduce magnitude of regression coefficients by choosing a parameter λ and minimizing

$$\frac{1}{2N}\sum_{n=1}^{N}\left[h_\theta\left(x^{(n)}\right)-y^{(n)}\right]^2+\lambda\sum_{n=1}^{N}b_i^2 \tag{4}$$

- This change has the effect of encouraging the model to keep the weights b_j as small as possibile;
- The Ridge regression should only be used for determining model parameters using the training set. Once the model parameters have been determined the penalty term should be removed for prediction;
- What happens as λ increases?

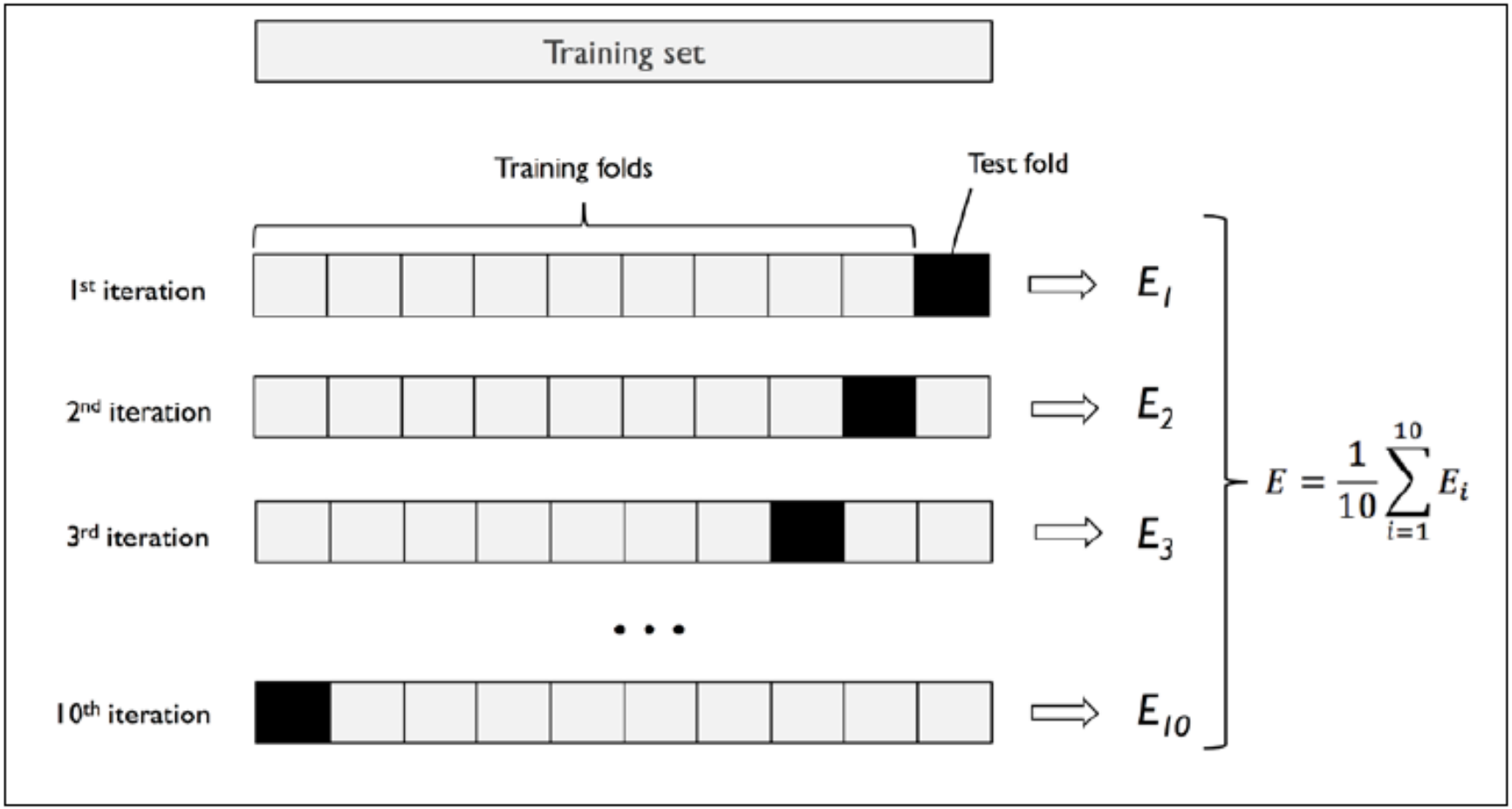
Lasso Regression

- Lasso is short for *Least Absolute Shrinkage and Selection Operator*;
- Similar to ridge regression except we minimize

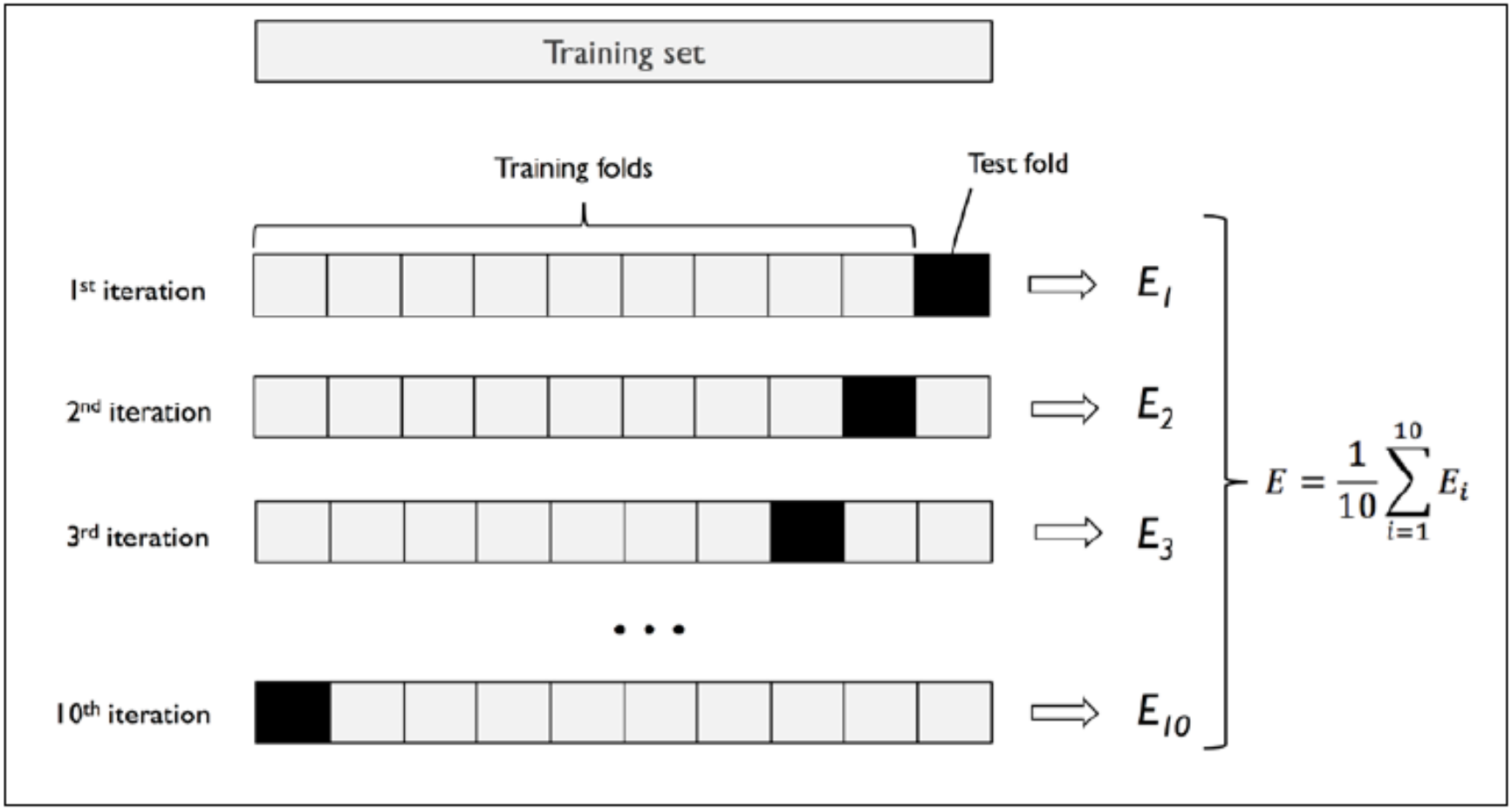
$$\frac{1}{2N}\sum_{n=1}^{N}\left[h_\theta\left(x^{(n)}\right)-y^{(n)}\right]^2+\lambda\sum_{n=1}^{N}|b_n| \tag{5}$$

- This function cannot be minimized analytically and so a variation on the gradient descent algorithm must be used;
- Lasso regression also has the effect of simplifying the model. It does this by setting the weights of unimportant features to zero. When there are a large number of features, Lasso can identify a relatively small subset of the features that form a good predictive model.

Lasso Regression: Geometrical Explanation



Lasso Regression: Geometrical Explanation



L1 Norm Constraint as a Cross Polytope & Level Sets of the Objective Function

- **L1 Norm Constraint as a Cross Polytope:**
  - In 2D, the L1 norm constraint $\|\beta\|_1 \leq t$ (for some $t > 0$) forms a diamond shape (a square rotated by 45 degrees) with four corners at $(\pm t, 0)$ and $(0, \pm t)$.
  - In higher dimensions, this shape generalizes to a cross polytope with $2n$ corners in n-dimensional space.
- **Level Sets of the Objective Function:**
  - In many Lasso explanations, we visualize level sets (contours) of the unconstrained objective—often ellipses in the case of squared-error loss—expanding outward from a central point (e.g., the ordinary least squares solution).
  - These ellipses will eventually intersect the L1 diamond or polytope.

Corners Promote Sparsity & Why It May Seem Less Obvious in 2D

Corners Promote Sparsity:

- The "sparsity" effect arises because corners of the L1 region correspond to solutions where one or more coefficients are zero (i.e., the solution "hits" an axis).
- In higher dimensions, there are many more corners (since there are $2n$ vertices in an n-dimensional cross polytope), which increases the likelihood that the outward-expanding contours will meet the L1 ball at a corner, thereby setting some coefficients to zero.

Corners Promote Sparsity & Why It May Seem Less Obvious in 2D

Why It May Seem Less Obvious in 2D:

- In two dimensions, the diamond only has four vertices. Depending on the orientation of the contours (ellipses), it may not always appear as "corner-favoring"
- Even in 2D, as you vary the penalty parameter t, you can often see the solution snap to a corner (meaning one coordinate goes to zero).
- In higher dimensions, this effect is magnified by the large number of corners, making it more likely that some coefficients become exactly zero.

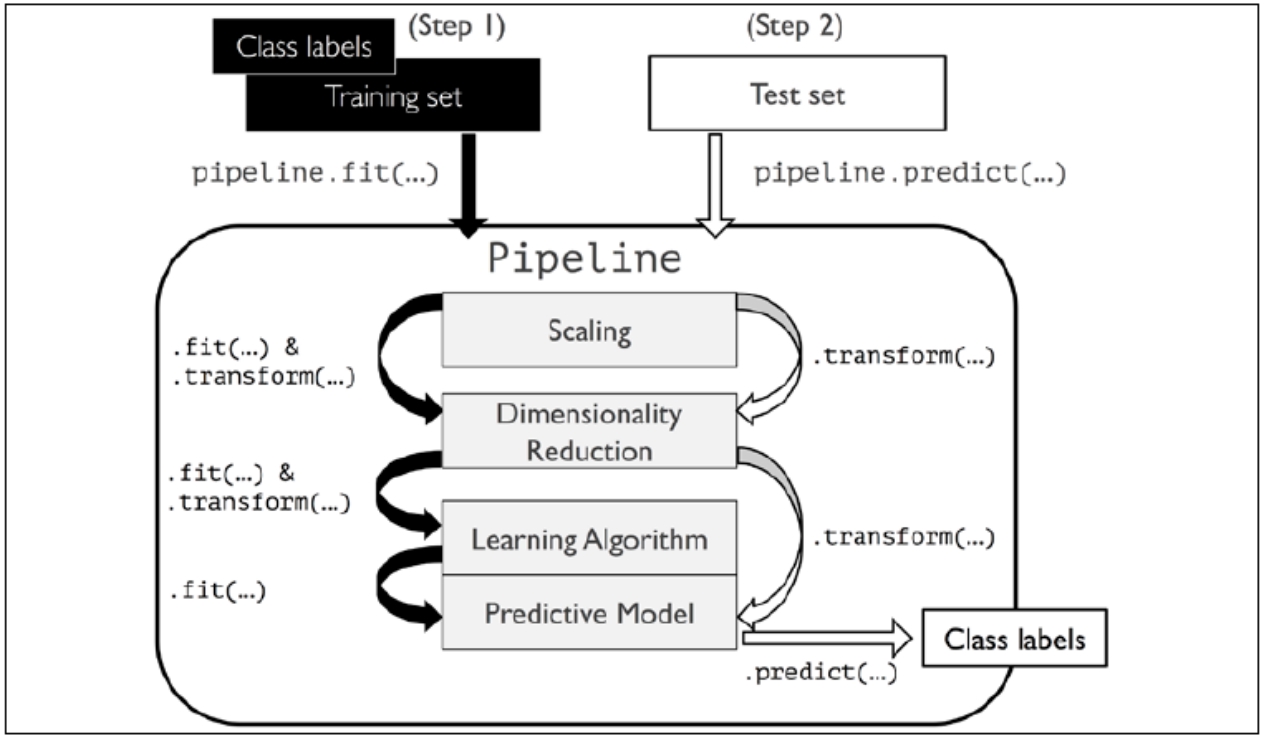
Elastic Net Regression (must use gradient descent)

- Middle ground between Ridge and Lasso
- Minimize

$$\frac{1}{2N}\sum_{n=1}^{N}\left[h_\theta\left(x^{(n)}\right)-y^{(n)}\right]^2+\lambda_1\sum_{n=1}^{N}b_n^2+\lambda_2\sum_{n=1}^{N}|b_n| \tag{6}$$

- In Lasso some weights are reduced to zero but others may be quite large. In Ridge, weights are small in magnitude but they are not reduced to zero. The idea underlying Elastic Net is that we may be able to get the best of both by making some weights zero while reducing the magnitude of the others.

Salary Vs Age Example: The Effect of Regularization



We apply regularization to the model:

$$Y = a + b_1 X + b_2 X^2 + b_3 X^3 + b_4 X^4 + b_5 X^5 \tag{7}$$

where Y is salary and X is age.

Salary Vs Age Example: The Effect of Regularization

Data with Z-score scaling

Observ.	X	X^2	X^3	X^4	X^5
1	−1.290	−1.128	−0.988	−0.874	−0.782
2	0.836	0.778	0.693	0.592	0.486
3	−1.148	−1.046	−0.943	−0.850	−0.770
4	−0.581	−0.652	−0.684	−0.688	−0.672
5	1.191	1.235	1.247	1.230	1.191
6	1.545	1.731	1.901	2.048	2.174
7	0.128	−0.016	−0.146	−0.253	−0.333
8	−0.227	−0.354	−0.449	−0.511	−0.544
9	0.482	0.361	0.232	0.107	−0.004
10	−0.936	−0.910	−0.861	−0.803	−0.745

Salary Vs Age Example: The Effect of Regularization

Ridge Results, $\lambda = 0.02$ is similar to quadratic model

λ	a	b_1	b_2	b_3	b_4	b_5
0	216.5	−32,623	135,403	−215,493	155,315	−42,559
0.02	216.5	97.8	36.6	−8.5	35.0	−44.6
0.10	216.5	56.5	28.1	3.7	−15.1	−28.4

Salary Vs Age Example: The Effect of Regularization

Lasso Results, $\lambda = 1$ is similar to the quadratic model

λ	a	b_1	b_2	b_3	b_4	b_5
0	216.5	−32,623	135,403	−215,493	155,315	−42,559
0.02	216.5	−646.4	2,046.6	0.0	−3,351.0	2,007.9
0.1	216.5	355.4	0.0	−494.8	0.0	196.5
1	216.5	147.4	0.0	0.0	−99.3	0.0

Let's code ...

Feature Selection

Introduction to Feature Selection

- Feature selection is the process of identifying and retaining only the most relevant variables for a machine learning model.
- The goal is to improve model performance by reducing noise and redundancy in the dataset.
- High-dimensional datasets often contain irrelevant or correlated features that do not contribute meaningful information for prediction.
- Unlike feature extraction, which transforms existing features into new ones (e.g., PCA, LDA), feature selection retains a subset of the original features.
- Feature selection enhances interpretability, reduces training time, and prevents overfitting, leading to a more efficient and effective model.

Why Feature Selection is Important?

- **Reduces Overfitting**: By removing irrelevant or redundant features, the model focuses only on meaningful variables, reducing the risk of learning noise instead of true patterns.
- **Improves Accuracy**: By eliminating uninformative features, the model generalizes better to unseen data, leading to improved prediction performance.
- **Speeds Up Training**: Fewer features mean lower computational complexity, reducing the time required to train and optimize machine learning models.

Why Feature Selection is Important?

- **Enhances Interpretability**: Models with fewer, well-selected features are easier to understand and explain, which is especially important in regulated industries such as finance and healthcare.
- **Mitigates Curse of Dimensionality**: High-dimensional data can negatively impact model performance due to sparsity; selecting relevant features helps mitigate this issue.
- **Facilitates Model Deployment**: A model with fewer variables is easier to implement in production environments, requiring less storage and computational resources.

How do we select features?

- A feature selection procedure combines a search technique with an evaluation method. The search technique proposes new feature subsets, and the evaluation measure determines the how good the subset is.
- In a perfect world, a feature selection method would evaluate all possible subsets of feature combinations and determine which one results in the best performing machine learning model.
- However, computational cost inhibits such a practice in reality. In addition, the optimal subset of features varies between machine learning models. A feature subset that optimizes one model‘s performance won‘t necessarily optimize another‘s.

Methods of Feature Selection

1. **Filter Methods**: These methods apply statistical techniques to evaluate the importance of each feature independently of the model. Common techniques include correlation analysis, Chi-square test, and mutual information.
2. **Wrapper Methods**: These methods iteratively evaluate subsets of features using machine learning models, selecting the best-performing combination. Examples include Recursive Feature Elimination (RFE), Forward Selection, and Backward Elimination.
3. **Embedded Methods**: These methods integrate feature selection within the model training process. Techniques such as Lasso (L1) regression and Decision Tree-based feature importance are commonly used.

Filter Methods

- A typical filter algorithm consists of two steps: it ranks features based on certain criteria and then chooses the highest-ranking features to train the machine learning models.
- Filter methods are generally univariate, so they rank each feature independently of the rest. Because of this, the filter methods tend to ignore any interactions that occur between features. Thus, redundant variables will not necessarily be eliminated by filter methods.
- However, some multivariate filter selection methods exist as well. These consider features in relations to others in the data set, making them naturally capable of handling redundant features. Their selection criteria scans for duplicated features and correlated features and provide simple but powerful methods to quickly remove redundant information.

Filter Methods

Constant, quasi-constant, and duplicated features

- The most basic and intuitive methods for feature selection consist of removing constant, quasi-constant, or duplicated features.
- Constant features only show one value for all the observations in the data set. That is, they show absolutely no variability.
- Quasi-constant features are similar; if most observations share the same value, then we'd label that feature quasi-constant. In practice, quasi-constant features typically refer to those variables where more than 95 to 99 percent of the observations show the same value.
- Duplicated features, as the name indicates, are those that are in essence, identical. That is, for every observation, they show the same value.

Filter Methods

Constant, quasi-constant, and duplicated features

- Although it sounds obvious and overly simple, many datasets contain a lot of constant, quasi-constant, and duplicated features.
- In fact, duplicated features often arise when generating new features by one-hot encoding of categorical variables.
- Removing these features is an easy but effective way to reduce the dimension of the feature space, without losing any significant information.

Filter Methods

Correlation

- Correlation measures the linear association between two or more variables.
- The higher the correlation, the more linearly associated the variables are.
- The central hypothesis is that good feature sets contain features that are highly correlated with the target, yet uncorrelated with each other.
- If two variables are correlated, we can predict one from the other.
- Therefore, if two features are correlated, the model only really needs one of them, as the second one does not add additional information.

Filter Methods

Mutual Information

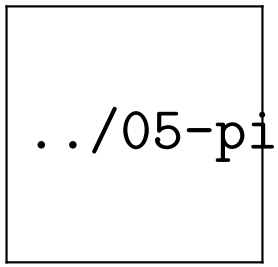
- Mutual information measures the mutual dependence between two variables, in this case, the feature and the target. Mutual information is similar to correlation, but more general; it doesn't strictly represent linear association. It measures how much knowing one of these variables reduces uncertainty in the other.

Fisher Score

- Fisher score uses the Chi-Square distribution to measure the dependency between two variables and works with categorical or discrete variables.
- Fisher score essentially compares the actual frequencies of the values of a variable with the expected frequencies if it had no relation to the target.

Wrapper Methods

- In comparison to filter methods, wrapper methods tend to be more computationally expensive, but select a better set of features.
- Wrapper methods use a specific machine learning algorithm in the selection process and choose the best subset of features tailored for that algorithm.
- This subset of features may not be optimal for a different machine learning model. In other words, **wrapper methods are not model agnostic**.



Example: RFE with Logistic Regression

Wrapper Methods

```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
rfe = RFE(model, n_features_to_select=5)
rfe.fit(X_train, y_train)
selected_features = X_train.columns[rfe.support_]
print("Selected Features:", selected_features)
```

Embedded Methods

- Feature selection occurs during model training.
- Uses model-specific techniques.
- Examples:
  - Lasso (L1) Regression
  - Decision Trees and Random Forests

Comparison of Methods

Method	Pros	Cons
Filter	Fast, scalable	Ignores interactions
Wrapper	Better selection	Expensive
Embedded	Model-optimized	Model-dependent

Understanding Pipelines in Scikit-Learn

- A pipeline is a sequence of data processing steps that are applied in a structured way.
- It helps automate machine learning workflows by chaining together multiple operations.
- Instead of manually applying transformations (e.g., scaling, feature selection, model training), a pipeline does it all in one step.
- Think of a pipeline as an assembly line:
  - Raw data enters the pipeline.
  - The data is preprocessed (e.g., missing values handled, scaling applied).
  - Feature selection picks the most relevant features.
  - A machine learning model is trained on the transformed data.
  - Predictions are generated efficiently.

Feature Selection Pipeline in Scikit-Learn

- Machine learning pipelines in Scikit-Learn allow seamless integration of preprocessing, feature selection, and model training.
- They enhance workflow efficiency, reproducibility, and ensure that transformations are applied consistently during training and inference.
- A typical pipeline consists of:
  - Preprocessing (e.g., standardization, encoding categorical variables)
  - Feature Selection (e.g., Recursive Feature Elimination, SelectKBest)
  - Model Training (e.g., Logistic Regression, Random Forest)
- Pipelines prevent data leakage by ensuring that feature selection and scaling are only learned from the training data.

Example: Feature Selection Pipeline in Scikit-Learn

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.linear_model import LogisticRegression

# Define the pipeline
pipeline = Pipeline([
# Standardize features
('scaler', StandardScaler()),
# Select top 5 features
('feature_selection', SelectKBest(score_func=f_classif, k=5)),
# Train logistic regression model
('classifier', LogisticRegression())
])
# Fit the pipeline on training data
pipeline.fit(X_train, y_train)
y_pred = pipeline.predict(X_test) # Make predictions
```

Advantages of Using Pipelines

- **Automation:** Reduces manual steps and minimizes human errors.
- **Prevents Data Leakage:** Ensures feature selection and scaling are applied correctly.
- **Code Reusability:** Simplifies experimentation with different models and preprocessing techniques.
- **Hyperparameter Tuning:** Easily integrates with GridSearchCV and RandomizedSearchCV for optimizing parameters.

Conclusion and Best Practices

- Choose method based on dataset size and complexity.
- Filter methods for quick insights.
- Wrapper methods for optimal selection.
- Embedded methods for model-specific optimization.
- Always validate selected features.

Dimensionality Reduction

Feature Selection: Follow a KISS Approach!

- As a dimensionality reduction technique, feature selection aims to choose a small subset of the relevant features from the original features by removing irrelevant, redundant, or noisy features.
- Feature selection usually can lead to better learning performance, higher learning accuracy, lower computational cost, and better model interpretability.
- The problem is important, because a high number of features in a dataset, comparable to or higher than the number of samples, leads to model overfitting, which in turn leads to poor results on the validation datasets.
- Additionally, constructing models from datasets with many features is more computationally demanding.

Feature Extraction and Feature Selection

- Even though feature extraction and feature selection processes share some overlap, often these terms are erroneously equated.
- **Feature extraction** is the process of using domain knowledge to extract new variables from raw data that make machine learning algorithms work.
- The **feature selection** process is based on selecting the most consistent, relevant, and non-redundant features.
- The feature selection process is based on selecting the most consistent, relevant, and non-redundant feature subset from feature vectors.
- It is not only reduces training time and model complexity, but it eventually helps to **prevent overfitting**.

Principal Component Analysis (PCA)

- PCA is a widely used technique that transforms the original features into a new set of orthogonal components.
- The first few principal components retain the most variance in the data.
- Helps in reducing dimensionality while preserving important patterns.
- PCA works by:
  - Standardizing the dataset.
  - Computing the covariance matrix.
  - Finding eigenvectors and eigenvalues.
  - Selecting the top principal components.
  - Transforming the data into the new feature space.
- PCA is effective for noise reduction and visualization.

Mathematics Behind PCA

- Given a dataset with n features, PCA computes the covariance matrix:

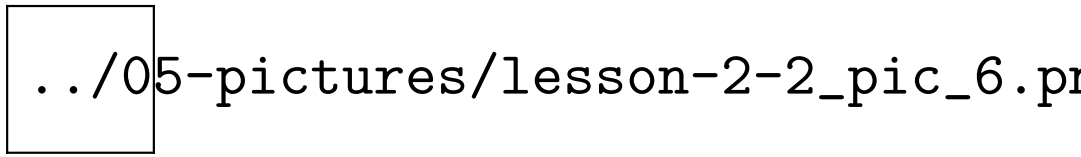
$$C = \frac{1}{n} X^T X \tag{8}$$

- The eigenvectors and eigenvalues of C represent the principal components and their importance.
- The data is projected onto the top k eigenvectors with the highest eigenvalues:

$$Z = XW \tag{9}$$

where W is the matrix of selected eigenvectors.

Standardization and Gradient Descent



Example: PCA in Scikit-Learn

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
```

Advantages and Limitations of PCA

- **Advantages:**
  - Reduces redundancy by removing correlated features.
  - Improves model efficiency by reducing computation time.
  - Helps in visualizing high-dimensional data.
- **Limitations:**
  - Assumes linear relationships in data.
  - Loses interpretability of original features.
  - Sensitive to data scaling; standardization is required.

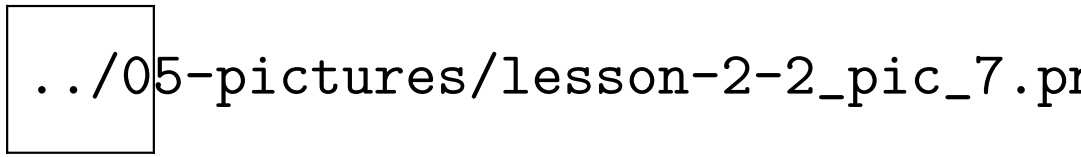
When to Use PCA?

- When dealing with high-dimensional data to reduce computational costs.
- When visualization of data is required in lower dimensions.
- When removing noise and redundancy is essential for better model performance.
- When feature interpretability is less important compared to efficiency.

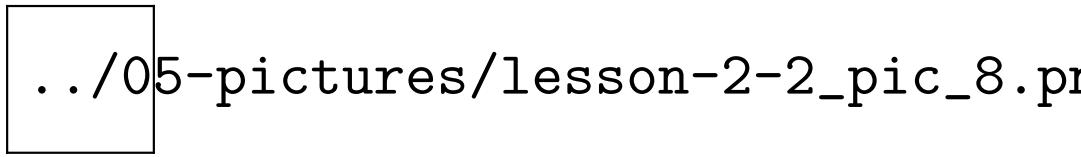
Linear Discriminant Analysis (LDA)

- LDA is a supervised dimensionality reduction technique that maximizes class separability.
- Unlike PCA, which focuses on variance, LDA optimizes for class separation.
- Works well for classification problems by finding a lower-dimensional space that retains discriminative information.
- LDA assumes that different classes generate data based on Gaussian distributions with the same covariance matrix.

LDA Vs PCA



LDA Vs PCA



Mathematics Behind LDA

- LDA computes two key matrices:
  - Within-class scatter matrix (S_W): Measures variance within each class.
  - Between-class scatter matrix (S_B): Measures variance between different classes.
- LDA solves the generalized eigenvalue problem:

$$S_W^{-1} S_B v = \lambda v \tag{10}$$

where eigenvectors corresponding to the largest eigenvalues define the optimal projection space.

Example: LDA in Scikit-Learn

```
from sklearn.discriminant_analysis
    import LinearDiscriminantAnalysis

from sklearn.preprocessing import StandardScaler

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply LDA
lda = LinearDiscriminantAnalysis(n_components=2)
X_lda = lda.fit_transform(X_scaled, y)
```

Advantages and Limitations of LDA

- **Advantages:**
  - Improves class separability in classification problems.
  - Reduces dimensionality while preserving discriminative information.
  - Computationally efficient compared to other non-linear methods.
- **Limitations:**
  - Assumes normally distributed classes with equal covariance matrices.
  - Less effective when class distributions overlap significantly.
  - Requires labeled data for training.

When to Use LDA?

- When working with classification tasks that require feature reduction.
- When the goal is to maximize class separation in lower-dimensional space.
- When the dataset follows Gaussian distribution assumptions.

Comparison: PCA vs LDA

Method	Objective	Type
PCA	Maximize variance	Unsupervised
LDA	Maximize class separability	Supervised

Conclusion

LDA

- LDA is a powerful tool for supervised dimensionality reduction, particularly in classification tasks.
- It optimizes feature space for class separability, making it useful for many real-world applications.
- Choosing between PCA and LDA depends on the problem: use PCA for general structure preservation and LDA for class separability.