Text Annotation Tasks:

1- Classify the entire document
2- Classify individual word tokens
3- Identify phrases “chunking”
4- Syntactic annotation (parsing)
5- Semantic annotation

Bi-LSTM

Bi-LSTM (Bi-directional Long Short-Term Memory) is a type of recurrent neural network (RNN) that processes input sequences in both forward and backward directions. It has two LSTM cells, one for processing the sequence forward in time and the other for processing the sequence backward in time. The output of the two LSTMs is then concatenated and used for downstream tasks such as classification or sequence generation.

Transformer

The Transformer is a neural network architecture for natural language processing tasks such as language translation, summarization, and language modeling. It is based on the idea of self-attention, which allows the model to directly attend to different positions in the input sequence rather than relying on recurrent connections or convolutions. This makes the Transformer much faster to train and more parallelizable than traditional models such as LSTM networks.

BERT

BERT (Bidirectional Encoder Representations from Transformers) is a deep learning model for natural language processing tasks developed by Google. It is based on the Transformer architecture and is trained using a large corpus of unannotated text. BERT is designed to pre-train deep bidirectional representations from unlabeled data by jointly conditioning on both left and right context in all layers. This allows BERT to achieve state-of-the-art performance on many natural language processing tasks.

*

The main difference between Bi-LSTM, Transformer, and BERT is the architecture they use to process input sequences. Bi-LSTM uses a recurrent architecture, the Transformer uses self-attention, and BERT uses a combination of both. BERT is also trained using a different method than the other two models, which allows it to achieve better performance on many natural language processing tasks.

Labeled Dependency Parsing

Labeled dependency parsing is a natural language processing task that involves analyzing the grammatical structure of a sentence and annotating each word with a dependency label indicating its grammatical role in the sentence. Dependency labels describe the relationships between words in a sentence, such as the subject of a verb or the object of a preposition.

For example, consider the following sentence:

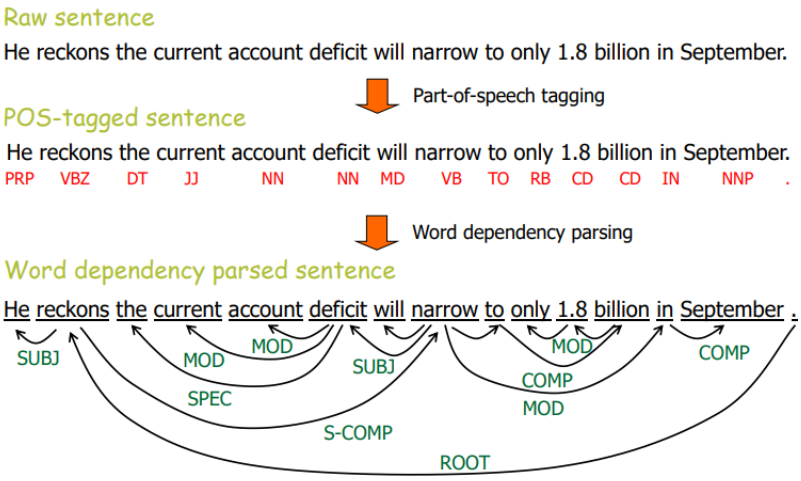
"The cat sat on the mat."

A labeled dependency parse of this sentence might look like this:

The cat sat on the mat

| | | | | |

det nsubj v case det n



Dependency Trees

A dependency tree is a tree-based representation of the grammatical structure of a sentence, where the nodes of the tree correspond to words in the sentence and the edges represent the dependencies between the words. The dependencies indicate the grammatical roles of the words in the sentence, such as the subject of a verb or the object of a preposition.

For example, consider the following sentence:

"The cat sat on the mat."

A dependency tree for this sentence might look like this:

sat

|

cat

|

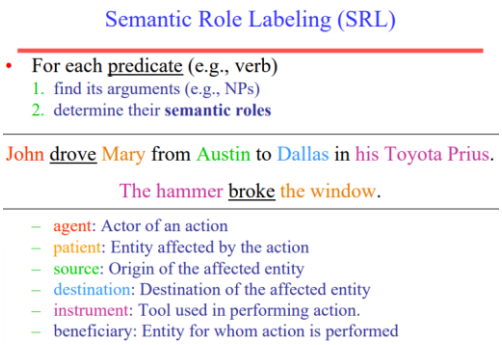
on

|

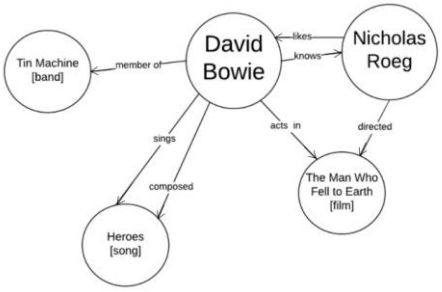
mat

In this example, "cat" is the subject of the verb "sat" and "mat" is the object of the preposition "on".

Semantic Role Labeling: For example, in the sentence "The cat chased the mouse," the verb is "chased," and the semantic roles would be "agent" (the cat) and "patient" (the mouse). The agent is the entity that is performing the action, and the patient is the entity that is affected by the action. Other common semantic roles include "instrument" (the object used to perform the action), "location" (the place where the action takes place), and "time" (the time when the action takes place).



The Semantic Web: A simple scheme for representing factual knowledge as a labeled graph.



Decoder and Encoder:

In natural language processing (NLP), an encoder is a component of a neural network that processes a sequence of input data and converts it into a fixed-length internal representation, known as a "latent representation." This internal representation is typically a lower-dimensional space that captures the important features of the input data and discards noise or unnecessary details.

The encoder is often followed by a decoder, which takes the latent representation produced by the encoder as input and generates a new sequence of data as output. The decoder can be thought of as a "reverse" encoder, which converts the latent representation back into a sequence of data that has a similar meaning to the input sequence.

One common application of encoders and decoders in NLP is in machine translation, where the encoder processes a sequence of words in one language (the input language) and the decoder generates a sequence of words in another language (the output language). The encoder and decoder

are trained together, using a large dataset of sentence pairs in the two languages, to learn how to translate between the languages.

Vector Space Model

In a vector space model (VSM) in natural language processing, each document is represented as a vector of identifiers such as index terms. The value of each dimension in the vector is the frequency with which the corresponding index term appears in the document. This representation allows for the analysis of relationships between a set of documents, such as calculating the similarity between two documents or identifying the most important terms in a set of documents.

For example, consider a set of three documents:

Doc 1: "the cat sat on the mat"

Doc 2: "the cat chased the mouse"

Doc 3: "the dog barked at the cat"

One possible set of index terms for these documents might be

{"the", "cat", "sat", "on", "mat", "chased", "mouse", "dog", "barked", "at"}

The vectors for these documents, using term frequency as the value for each dimension, might look like this:

Doc 1: [2, 1, 1, 1, 1, 0, 0, 0, 0, 0]

Doc 2: [2, 1, 0, 0, 0, 1, 1, 0, 0, 0]

Doc 3: [1, 1, 0, 0, 0, 0, 0, 1, 1, 1]

These vectors can then be used to calculate the similarity between the documents, identify the most important terms, or perform other types of analysis.

Vector Embedding of Words

Traditional Method - Bag of Words Model

- Either uses one hot encoding.
  - Each word in the vocabulary is represented by one bit position in a HUGE vector.
  - For example, if we have a vocabulary of 10000 words, and "Hello" is the 4th word in the dictionary, it would be represented by: 0 0 0 1 0 0 . . . . . 0 0 0
- Or uses document representation.
  - Each word in the vocabulary is represented by its presence in documents.
  - For example, if we have a corpus of 1M documents, and "Hello" is in 1th, 3th and 5th documents ***only***, it would be represented by: 1 0 1 0 1 0 . . . . . 0 0 0
- Context information is not utilized.

Word Embeddings

Stores each word in as a point in space, where it is represented by a dense vector of fixed number of dimensions (generally 300) .

Unsupervised, built just by reading huge corpus.

For example, "Hello" might be represented as : [0.4, -0.11, 0.55, 0.3 . . . 0.1, 0.02].

Dimensions are basically projections along different axes, more of a mathematical concept.

Word2Vec: Context Representation Models

In the CBOW model, the goal is to predict the target word given the context words. The input to the model is a window of context words surrounding the target word, and the output is the predicted target word. For example, given the input context "the cat sat on", the model might predict the target word "mat" as the output.

In the skip-gram model, the goal is to predict the context words given the target word. The input to the model is a single target word, and the output is the predicted context words. For example, given the input target word "cat", the model might predict the context words "the" and "sat on" as the output.

Limitations of Word2vec

Out of Vocabulary(OOV) Words: In Word2Vec, an embedding is created for each word. As such, it can't handle any words it has not encountered during its training.

Morphology: For words with same radicals such as "eat" and "eaten", Word2Vec doesn't do any parameter sharing. Each word is learned uniquely based on the context it appears in. Thus, there is scope for utilizing the internal structure of the word to make the process more efficient.

FastText

word2vec treats each word in corpus like an atomic entity and generates a vector for each word. In this sense Word2vec is very much like Glove — both treat words as the smallest unit to train on.

FastText (which is essentially an extension of word2vec model), treats each word as composed of character ngrams. So the vector for a word is made of the sum of this character n grams. For example, the word vector "apple" is a sum of the vectors of the n-grams:

"<ap", "app", "appl", "apple", "apple>", "ppl", "pple", "pple>", "ple", "ple>", "le>"

The key difference between FastText and Word2Vec is the use of n-grams.

GloVe (Global Vectors)

GloVe works by training a shallow neural network to predict the co-occurrence counts of words in a large corpus. It uses the co-occurrence counts to learn the word vectors such that the dot product of the vectors of two words is proportional to the logarithm of the co-occurrence count of the words. This allows GloVe to capture both the syntactic and semantic relationships between words.

ElMo

Language is complex, and context can completely change the meaning of a word in a sentence. Need a model which captures the different nuances of the meaning of words given the surrounding text. Previous models (GloVe, Word2Vec, etc.) only have one representation per word. They can't capture these ambiguities. When you only have one representation, all levels of meaning are combined.

OUTLIERS

Outliers are observations in a dataset that are significantly different from the majority of the data. They can be caused by a variety of factors such as measurement errors, experimental errors, or data entry errors. Outliers can have a significant impact on the results of statistical analyses and can distort the overall pattern in the data.

Noisy data, on the other hand, refers to data that is corrupted by random errors or variation. Noisy data can be caused by a variety of factors such as measurement errors, environmental factors, or hardware errors. Noisy data can also affect the results of statistical analyses and make it more difficult to detect patterns in the data.

The main difference between outliers and noisy data is that outliers are extreme values that are significantly different from the majority of the data, while noisy data is characterized by random errors or variations that affect the accuracy of the data. Outliers can be identified by visualizing the data or using statistical tests, while noisy data can be reduced by applying data cleaning and preprocessing techniques such as smoothing or denoising.

Noise should be removed before outlier detection

Types of outliers

Global Outlier

Bir dataset içerisinde diğer verilere göre çok ayrı olanlardır. Örnek olarak sıcaklık verilerinde 100 derece olması gibi

Contextual Outlier

Bir dataset içerisinde bir context olan veriler arasında ayrı olanlardır. Mesela sıcaklık verilerinde Antalya'nın -10 derece olması gibi

Collective Outlier

Bir dataset içerisinde ayrı ayrı olarak bakıldığında outlier olmayan ama bir araya geldiklerinde outlier olan durumlardır.

ÖRNEK:

A fist-size meteorite impacting a house in your neighborhood is a **global** outlier because it's a truly rare event that meteorites hit buildings. Your neighborhood getting buried in two feet of snow would be a **contextual** outlier if the snowfall happened in the middle of summer and you normally don't get any snow outside of winter. Every one of your neighbors moving out of the neighborhood on the same day is a **collective** outlier because although it's definitely not rare that people move from one residence to the next, it is very unusual that an entire neighborhood relocates at the same time.

Challenges of Outlier Detection

- Normal ve outlierlar arasında gri alan bulunması (Bütün normalleri bulmanın zor olması)
- Uygulamalara göre aralığın değişmesi (sağlık için küçük değerler outlier olabilirken finans için büyük fluctuationslar etkili olur)
- Noisy datanın normal datayı etkiliyerek outlierları saklaması
- Outlierları bulduktan sonra bunların neden outlier olduklarını anlamak (Bulunan outlierları justificate etmek için)

Outlier Detection Methods

- Based on whether user-labeled examples of outliers can be obtained:
    - Supervised, semi-supervised vs unsupervised methods
- Based on assumptions about normal data and outliers:
    - Statistical, proximity-based, clustering-based methods

Supervised Methods:

Supervised outlier detection methods are those that require labeled training data, in which the normal and outlier objects are already known. The goal of these methods is to learn a model that can identify these objects in new, unseen data. Some examples of supervised outlier detection methods include:

Decision tree-based methods: These methods build a decision tree model using the training data, and then use the tree to classify new objects as either normal or outlier.

Support vector machines (SVMs): These methods try to find a hyperplane in the feature space that maximally separates the normal objects from the outlier objects.

Neural networks: These methods learn a nonlinear model of the data using an artificial neural network, and then use the model to classify new objects as either normal or outlier.

Logistic regression: These methods model the probability that an object is an outlier as a function of its features, and then use the model to classify new objects as either normal or outlier.

Ensemble methods: These methods combine the predictions of multiple different models to make a final outlier prediction. Examples include random forests and boosting algorithms.

Challenges are:

Imbalanced classes, i.e., outliers are rare: Boost the outlier class and make up some artificial outliers

Catch as many outliers as possible, i.e., recall is more important than accuracy (i.e., not mislabeling normal objects as outliers)

Unsupervised Methods:

Assume the normal objects are somewhat ``clustered'' into multiple groups, each having some distinct features

An outlier is expected to be far away from any groups of normal objects

Weakness: Cannot detect collective outlier effectively

Semi-supervised methods:

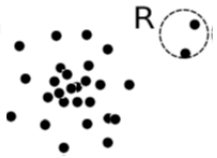
Semi-supervised learning can be a useful technique for improving the performance of machine learning models when only a small amount of labeled data is available

Statistical Methods:

Assume that the normal data follow some statistical model. The data not following the model are outliers.

Example (right figure): First use Gaussian distribution to model the normal data

- For each object y in region R, estimate $g_D(y)$, the probability of y fits the Gaussian distribution
- If $g_D(y)$ is very low, y is unlikely generated by the Gaussian model, thus an outlier



Effectiveness of statistical methods: highly depends on whether the assumption of statistical model holds in the real data

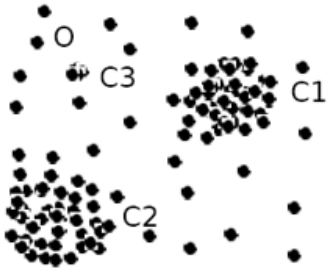
(Z-score method)

They are divided into 2 groups:

1- Parametric: Parametric methods for outlier detection are methods that assume that the data follows a certain distribution or pattern, and use this assumption to identify points in the data that are significantly different from the other points. These methods involve fitting a parametric model to the data, and using the parameters of the model to identify outliers.
2- Non-parametric methods for outlier detection, on the other hand, do not make any assumptions about the distribution of the data, and are generally more flexible and robust to deviations from the assumed distribution. Examples of non-parametric methods for outlier detection include the local outlier factor (LOF) algorithm and the density-based spatial clustering of applications with noise (DBSCAN) algorithm, which are both based on the density of points in the data rather than on any specific distribution.
   Overall, non-parametric methods can be more robust to deviations from the assumed distribution, but may be less interpretable and may require more computational resources to implement.

Detection of Multivariate outliers => Transform the multivariate outlier detection task into a univariate outlier detection problem then solve.

- Assuming data generated by a normal distribution could be sometimes overly simplified
- Example (right figure): The objects between the two clusters cannot be captured as outliers since they are close to the estimated mean

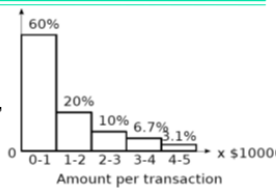


- To overcome this problem, assume the normal data is generated by two normal distributions. For any object o in the data set, the probability that o is generated by the mixture of the two distributions is given by

$$Pr(o|\Theta_1, \Theta_2) = f_{\Theta_1}(o) + f_{\Theta_2}(o)$$

Non-Parametric Methods: Detection Using Histogram

- The model of normal data is learned from the input data without any *a priori* structure.
- Often makes fewer assumptions about the data, and thus can be applicable in more scenarios
- Outlier detection using histogram:
  - Figure shows the histogram of purchase amounts in transactions
  - A transaction in the amount of $7,500 is an outlier, since only 0.2% transactions have an amount higher than $5,000
- Problem: Hard to choose an appropriate bin size for histogram
  - Too small bin size → normal objects in empty/rare bins, false positive
  - Too big bin size → outliers in some frequent bins, false negative
- Solution: Adopt kernel density estimation to estimate the probability density distribution of the data. If the estimated density function is high, the object is likely normal. Otherwise, it is likely an outlier.



Kernel density estimation (KDE) is a non-parametric method for estimating the probability density function of a random variable. It involves using a kernel function to smooth the data and estimate the underlying density function.

One way to use KDE for outlier detection is to estimate the density function of the data and then identify points that have a significantly lower density than the other points. These points can be considered outliers, since they have a lower probability of belonging to the normal distribution of the data.

Proximity-Based Methods:

An object is an outlier if the nearest neighbors of the object are far away, i.e., the **proximity** of the object is **significantly deviates** from the proximity of most of the other objects in the same data set

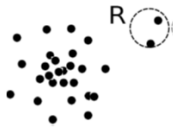
The effectiveness of proximity-based methods highly relies on the proximity measure

Often have a difficulty in finding a group of outliers which stay close to each other

Two major types of proximity-based outlier detection: Distance-based vs. density-based

Example (right figure): Model the proximity of an object using its 3 nearest neighbors

- Objects in region R are substantially different from other objects in the data set.
- Thus the objects in R are outliers



(Knn could be an example)

1- Distance-based outlier detection: An object o is an outlier if its neighborhood does not have enough other points
2- Density-based outlier detection: An object o is an outlier if its density is relatively much lower than that of its neighbors
The LOF algorithm measures the local density of a point in the data, and compares it to the densities of other points in the data. Points that have a much lower density than their neighbors are considered outliers.

Clustering-Based Methods:

Normal data belong to large and dense clusters, whereas outliers belong to small or sparse clusters, or do not belong to any clusters . Since there are many clustering methods, there are many clustering-based outlier detection methods as well

Clustering is expensive: straightforward adaption of a clustering method for outlier detection can be costly and does not scale up well for large data sets

Example (right figure): two clusters

- All points not in R form a large cluster
- The two points in R form a tiny cluster, thus are outliers

(DBSCAN and K-means could be example)

An object is an outlier if (1) it does not belong to any cluster, (2) there is a large distance between the object and its closest cluster , or (3) it belongs to a small or sparse cluster

Case I: Not belong to any cluster

Using a density-based clustering method such as DBSCAN

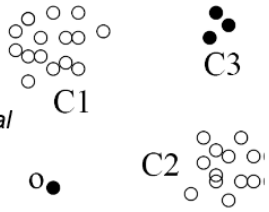
Case 2: Far from its closest cluster

Using k-means, partition data points of into clusters. For each object o, assign an outlier score based on its distance from its closest center. If dist(o, co)/avg_dist(co) is large, likely an outlier.

Case 3: Detecting outliers in small clusters

FindCBLOF: Detect outliers in small clusters

- Find clusters, and sort them in decreasing size
- To each data point, assign a *cluster-based local outlier factor* (CBLOF):
- If obj p belongs to a large cluster, CBLOF = cluster_size X similarity between p and cluster
- If p belongs to a small one, CBLOF = cluster size X similarity betw. p and the closest large cluster



Ex. In the figure, o is outlier since its closest large cluster is C_1, but the similarity between o and C_1 is small. For any point in C_3, its closest large cluster is C_2 but its similarity from C_2 is low, plus $|C_3| = 3$ is small

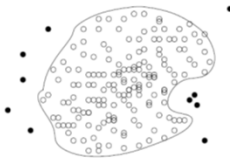
Clustering-Based Method: Strength and Weakness

- Strength
  - Detect outliers without requiring any labeled data
  - Work for many types of data
  - Clusters can be regarded as summaries of the data
  - Once the cluster are obtained, need only compare any object against the clusters to determine whether it is an outlier (fast)
- Weakness
  - Effectiveness depends highly on the clustering method used—they may not be optimized for outlier detection
  - High computational cost: Need to first find clusters
  - A method to reduce the cost: Fixed-width clustering
    - A point is assigned to a cluster if the center of the cluster is within a pre-defined distance threshold from the point
    - If a point cannot be assigned to any existing cluster, a new cluster is created and the distance threshold may be learned from the training data under certain conditions

Classification-based Methods

Classification-Based Method I: One-Class Model

- Idea: Train a classification model that can distinguish “normal” data from outliers
- A brute-force approach: Consider a training set that contains samples labeled as “normal” and others labeled as “outlier”
  - But, the training set is typically heavily biased: # of “normal” samples likely far exceeds # of outlier samples
  - Cannot detect unseen anomaly



- One-class model: A classifier is built to describe only the normal class.
  - Learn the decision boundary of the normal class using classification methods such as SVM
  - Any samples that do not belong to the normal class (not within the decision boundary) are declared as outliers
  - Adv: can detect new outliers that may not appear close to any outlier objects in the training set
  - Extension: Normal objects may belong to multiple classes

One type of classification-based outlier detection model is the one-class model. A one-class model is a binary classification model that is trained to distinguish between a single class (the normal class) and all other points (the outlier class). It is typically used in situations where there is a large amount of normal data available, but only a small amount of outlier data.

To train a one-class model, the model is typically fit to the normal data using an unsupervised learning algorithm, such as clustering or density estimation. The model is then used to predict whether new points are normal or outlier points.

Classification-Based Method II: Semi-Supervised Learning

- Semi-supervised learning: Combining classification-based and clustering-based methods
- Method
  - Using a clustering-based approach, find a large cluster, C, and a small cluster, C_1
  - Since some objects in C carry the label "normal", treat all objects in C as normal
  - Use the one-class model of this cluster to identify normal objects in outlier detection
  - Since some objects in cluster C_1 carry the label "outlier", declare all objects in C_1 as outliers
  - Any object that does not fall into the model for C (such as *a*) is considered an outlier as well

- Comments on classification-based outlier detection methods
  - Strength: Outlier detection is fast
  - Bottleneck: Quality heavily depends on the availability and quality of the training set, but often difficult to obtain representative and high-quality training data

Mining Contextual Outliers I: Transform into Conventional Outlier Detection

If the contexts can be clearly identified, transform it to conventional outlier detection

1. Identify the context of the object using the contextual attributes
2. Calculate the outlier score for the object in the context using a conventional outlier detection method

Ex. Detect outlier customers in the context of customer groups

- Contextual attributes: *age group, postal code*
- Behavioral attributes: *# of trans/yr*, *annual total trans. amount*

Steps: (1) locate c's context, (2) compare c with the other customers in the same group, and (3) use a conventional outlier detection method

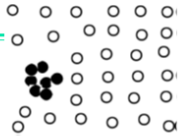
If the context contains very few customers, generalize contexts

- Ex. Learn a mixture model U on the contextual attributes, and another mixture model V of the data on the behavior attributes
- Learn a mapping $p(V_i|U_j)$: the probability that a data object o belonging to cluster U_j on the contextual attributes is generated by cluster Vi on the behavior attributes
- Outlier score: $S(\boldsymbol{o}) = \sum_{U_j} p(\boldsymbol{o} \in U_j) \sum_{V_i} p(\boldsymbol{o} \in V_i) p(V_i|U_j)$

Mining Contextual Outliers II: Modeling Normal Behavior with Respect to Contexts

- In some applications, one cannot clearly partition the data into contexts
  - Ex. if a customer suddenly purchased a product that is unrelated to those she recently browsed, it is unclear how many products browsed earlier should be considered as the context
- Model the "normal" behavior with respect to contexts
  - Using a training data set, train a model that predicts the expected behavior attribute values with respect to the contextual attribute values
  - An object is a contextual outlier if its behavior attribute values significantly deviate from the values predicted by the model
- Using a prediction model that links the contexts and behavior, these methods avoid the explicit identification of specific contexts
- Methods: A number of classification and prediction techniques can be used to build such models, such as regression, Markov Models, and Finite State Automaton

Mining Collective Outliers I: On the Set of "Structured Objects"



- Collective outlier if objects as a group deviate significantly from the entire data
- Need to examine the *structure* of the data set, i.e, the relationships between multiple data objects
- Each of these structures is inherent to its respective type of data
  - For temporal data (such as time series and sequences), we explore the structures formed by time, which occur in segments of the time series or subsequences
  - For spatial data, explore local areas
  - For graph and network data, we explore subgraphs
- Difference from the contextual outlier detection: the structures are often not explicitly defined, and have to be discovered as part of the outlier detection process.
- Collective outlier detection methods: two categories
  - Reduce the problem to conventional outlier detection
    - Identify *structure units*, treat each structure unit (e.g., subsequence, time series segment, local area, or subgraph) as a data object, and extract features
    - Then outlier detection on the set of "structured objects" constructed as such using the extracted features

Collective outliers are groups of points in a dataset that are unusual or unexpected when considered together, but may not be unusual or unexpected when considered individually. There are several approaches to detecting collective outliers, including:

Reducing the problem to conventional outlier detection: This approach involves identifying the structures in the data that are relevant to the specific context of the study, and then treating each structure unit (e.g., a subsequence in a time series, a local area in spatial data, or a subgraph in graph data) as a single data object. Features can be extracted from these structured objects, and then traditional outlier detection methods can be applied to the set of structured objects using the extracted features.

Identifying structure units and extracting features: This approach involves identifying the structures in the data that are relevant to the specific context of the study, and then extracting features from each structure unit. Outlier detection can then be performed on the set of structured objects using the extracted features.

Collective outlier detection is different from contextual outlier detection in that the structures in the data are often not explicitly defined, and have to be discovered as part of the outlier detection process. This can be a complex task that requires careful analysis of the data and a thorough understanding of the specific context and goals of the study.

Mining Collective Outliers II: Direct Modeling of the Expected Behavior of Structure Units

- Models the expected behavior of structure units directly
- Ex. 1. Detect collective outliers in online social network of customers
  - Treat each possible subgraph of the network as a structure unit
  - Collective outlier: An *outlier subgraph* in the social network
    - Small subgraphs that are of very low frequency
    - Large subgraphs that are surprisingly frequent
- Ex. 2. Detect collective outliers in temporal sequences
  - Learn a Markov model from the sequences
  - A subsequence can then be declared as a collective outlier if it significantly deviates from the model
- Collective outlier detection is subtle due to the challenge of exploring the structures in data
  - The exploration typically uses heuristics, and thus may be application dependent
  - The computational cost is often high due to the sophisticated mining process

Challenges for outlier detection in High-Dimensional Data

Challenges for Outlier Detection in High-Dimensional Data

- Interpretation of outliers
  - Detecting outliers without saying why they are outliers is not very useful in high-D due to many features (or dimensions) are involved in a high-dimensional data set
  - E.g., which subspaces that manifest the outliers or an assessment regarding the “outlier-ness” of the objects
- Data sparsity
  - Data in high-D spaces are often sparse
  - The distance between objects becomes heavily dominated by noise as the dimensionality increases
- Data subspaces
  - Adaptive to the subspaces signifying the outliers
  - Capturing the local behavior of data
- Scalable with respect to dimensionality
  - # of subspaces increases exponentially

Approach I: Extending Conventional Outlier Detection

- Method 1: Detect outliers in the full space, e.g., HilOut Algorithm
  - Find distance-based outliers, but use the ranks of distance instead of the absolute distance in outlier detection
  - For each object *o*, find its k-nearest neighbors: $nn_1(o)$, . . . , $nn_k(o)$
  - The weight of object o: $w(\boldsymbol{o}) = \sum_{i=1}^{k} dist(\boldsymbol{o}, nn_i(\boldsymbol{o}))$
  - All objects are ranked in weight-descending order
  - Top-*l* objects in weight are output as outliers (*l*: user-specified parm)
  - Employ space-filling curves for approximation: scalable in both time and space w.r.t. data size and dimensionality
- Method 2: Dimensionality reduction
  - Works only when in lower-dimensionality, normal instances can still be distinguished from outliers
  - PCA: Heuristically, the principal components with low variance are preferred because, on such dimensions, normal objects are likely close to each other and outliers often deviate from the majority

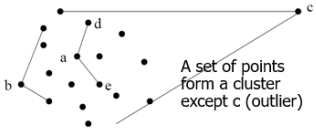
Approach II: Finding Outliers in Subspaces

- Extending conventional outlier detection: Hard for outlier interpretation
- Find outliers in much lower dimensional subspaces: easy to interpret *why* and *to what extent* the object is an outlier
  - E.g., find outlier customers in certain subspace: *average transaction amount* >> *avg.* and *purchase frequency* << avg.
- Ex. A grid-based subspace outlier detection method
  - Project data onto various subspaces to find an area whose density is much lower than average
  - Discretize the data into a grid with φ equi-depth (why?) regions
  - Search for regions that are significantly sparse
    - Consider a k-d cube: k ranges on k dimensions, with n objects
    - If objects are independently distributed, the expected number of objects falling into a k-dimensional region is $(1/\varphi)^k n = f^k n$, the standard deviation is $\sqrt{f^k(1-f^k)n}$
    - The sparsity coefficient of cube C: $S(C) = \frac{n(C) - f^k n}{\sqrt{f^k(1-f^k)n}}$
    - If S(C) < 0, C contains less objects than expected
    - The more negative, the sparser C is and the more likely the objects in C are outliers in the subspace

42

Approach III: Modeling High-Dimensional Outliers

- Develop new models for high-dimensional outliers directly
- Avoid proximity measures and adopt new heuristics that do not deteriorate in high-dimensional data



- Ex. Angle-based outliers: Kriegel, Schubert, and Zimek [KSZ08]
- For each point o, examine the angle ∠xoy for every pair of points x, y.
  - Point in the center (e.g., a), the angles formed differ widely
  - An outlier (e.g., c), angle variable is substantially smaller
- Use the variance of angles for a point to determine outlier
- Combine angles and distance to model outliers
  - Use the distance-weighted angle variance as the outlier score
  - Angle-based outlier factor (ABOF): $ABOF(\boldsymbol{o}) = VAR_{\boldsymbol{x},\boldsymbol{y} \in D, \boldsymbol{x} \neq \boldsymbol{o}, \boldsymbol{y} \neq \boldsymbol{o}} \frac{\langle \overrightarrow{\boldsymbol{ox}}, \overrightarrow{\boldsymbol{oy}} \rangle}{dist(\boldsymbol{o},\boldsymbol{x})^2 dist(\boldsymbol{o},\boldsymbol{y})^2}$
  - Efficient approximation computation method is developed
  - It can be generalized to handle arbitrary types of data

43

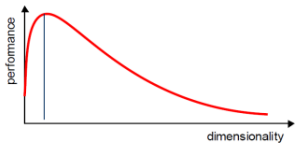
Dimensionality Reduction

Increasing the number of features will not always improve classification accuracy.

The number of training examples required increases exponentially with dimensionality d (i.e., k to the d).

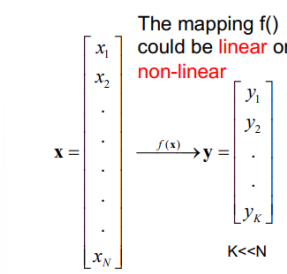
What is the objective?

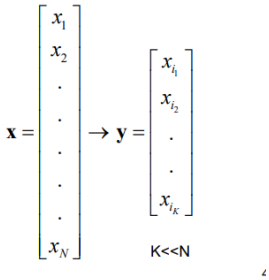
- Choose an optimum set of features of lower dimensionality to improve classification accuracy.



Feature extraction: finds a set of new features (i.e., through some mapping f()) from the existing features.

Feature selection: chooses a subset of the original features.



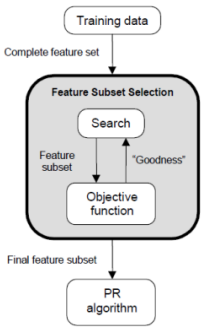


Feature Selection

Given a set of **N** features, the goal of **feature selection** is to select a subset of **K** features (**K << N**) in order to minimize the classification error.

Feature selection is an **optimization** problem.

- **Step 1:** Search the space of possible feature subsets.
- **Step 2:** Pick the subset that is optimal or near-optimal with respect to some objective function.



Search methods

- Exhaustive
- Heuristic
- Randomized

Evaluation methods

- Filter
- Wrapper

Search Methods

- Assuming n features, an exhaustive search would require examining $\binom{n}{d}$ possible subsets of size d.
- The number of subsets grows combinatorially, making exhaustive search impractical.
  - e.g., exhaustive evaluation of 10 out of 20 features involves 184,756 feature subsets.
  - e.g., exhaustive evaluation of 10 out of 100 involves more than 10^{13} feature subsets.
- In practice, heuristics are used to speed-up search but they **cannot** guarantee optimality.

Filter

- Evaluation is **independent** of the classification algorithm.
- The objective function is based on the information content of the feature subsets, e.g.:
  - interclass distance
  - statistical dependence
  - information-theoretic measures (e.g., mutual information).

Wrapper

- Evaluation uses criteria **related** to the classification algorithm.
- The objective function is based on the predictive accuracy of the classifier, e.g.,:
  - recognition accuracy on a "validation" data set.

Filter methods evaluate the relevance of each feature to the response variable, independently of the model that is being used. These methods rely on statistical tests or other heuristics to assess the importance of each feature. Some examples of filter methods are the Chi-squared test, the Mutual Information criterion, and the ANOVA F-value.

Wrapper methods, on the other hand, evaluate the performance of a subset of features by training a model using that subset and then evaluating its performance. These methods are more computationally expensive than filter methods, as they require training multiple models for different subsets of features. Some examples of wrapper methods are forward selection, backward elimination, and recursive feature elimination.

Wrapper Methods

- Advantages
  - Achieve higher recognition accuracy compared to filter methods since they use the classifier itself in choosing the optimum set of features.
- Disadvantages
  - Much slower compared to filter methods since the classifier must be trained and tested for each candidate feature subset.
  - The optimum feature subset might not work well for other classifiers.

Filter Methods

- Advantages
  - Much faster than wrapper methods since the objective function has lower computational requirements.
  - The optimum feature set might work well with various classifiers as it is not tied to a specific classifier.
- Disadvantages
  - Achieve lower recognition accuracy compared to wrapper methods.
  - Have a tendency to select more features compared to wrapper methods.

Naïve Search

- Given n features:
  1. Sort them in decreasing order of their "goodness" (i.e., based on some objective function).
  2. Choose the top d features from this sorted list.

- Disadvantages
  - Correlation among features is not considered.
  - The best pair of features may not even contain the best individual feature.

Sequential forward selection (SFS)

(heuristic search)

First, the best **single** feature is selected (i.e., using some objective function).

Then, **pairs** of features are formed using one of the remaining features and this best feature, and the best pair is selected.

Next, **triplets** of features are formed using one of the remaining features and these two best features, and the best triplet is selected.

This procedure continues until a predefined number of features are selected.

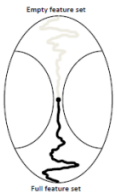
Sequential backward selection (SBS)

(heuristic search)

- First, the objective function is computed for all **n** features.
- Then, each feature is deleted one at a time, the objective function is computed for all subsets with **n-1** features, and the worst feature is discarded.
- Next, each feature among the remaining **n-1** is deleted one at a time, and the worst feature is discarded to form a subset with **n-2** features.
- This procedure continues until a predefined number of features are left.

Bidirectional Search (BDS)

- BDS applies SFS and SBS simultaneously:
  - SFS is performed from the empty set.
  - SBS is performed from the full set.
- To guarantee that SFS and SBS converge to the same solution:
  - Features already selected by SFS cannot be removed by SBS.
  - Features already removed by SBS cannot be added by SFS.



1. Start SFS with $Y_F = \{\emptyset\}$
2. Start SBS with $Y_B = X$
3. Select the best feature

$$x^+ = \arg\max_{x \in Y_{F_k}, x \in F_{B_k}} J(Y_{F_k} + x)$$

$$Y_{F_{k+1}} = Y_{F_k} + x^+$$

4. Remove the worst feature

$$x^- = \arg\max_{x \in Y_{B_k}, x \in Y_{F_{k+1}}} J(Y_{B_k} - x)$$

$$Y_{B_{k+1}} = Y_{B_k} - x^-; \; k = k +$$

5. Go to 2

Limitations of SFS and SBS

- The main limitation of SFS is that it is unable to remove features that become non useful after the addition of other features.
- The main limitation of SBS is its inability to reevaluate the usefulness of a feature after it has been discarded.

Limitations of SFS

It is computationally expensive: As a wrapper method, forward selection requires training and evaluating multiple models for different subsets of features. This can be time-consuming and may not be practical for very large datasets.

It may not always find the optimal subset of features: Forward selection is a greedy algorithm that makes locally optimal decisions at each step. This means that it may not find the globally optimal subset of features that leads to the best overall model performance.

It may over-fit the data: If the number of features is significantly increased, the model may end up fitting the training data too well, leading to overfitting and poor generalization to new data.

It does not consider interactions between features: Forward selection only evaluates the performance of individual features and does not consider interactions between features. This means that it may include features that are not important on their own, but are important in combination with other features

It is sensitive to the choice of the initial model: The performance of the final model obtained through forward selection can be heavily influenced by the initial model that is used. If the initial model is not sufficiently powerful, the algorithm may not be able to select important features that are relevant to the response variable.

Limitations of SBS

It is computationally expensive: As a wrapper method, backward elimination requires training and evaluating multiple models for different subsets of features. This can be time-consuming and may not be practical for very large datasets.

It is sensitive to the choice of the initial model: The performance of the final model obtained through backward elimination can be heavily influenced by the initial model that is used. If the initial model is not sufficiently powerful, the algorithm may eliminate important features that are actually relevant to the response variable.

It may not always find the optimal subset of features: Backward elimination is a greedy algorithm that makes locally optimal decisions at each step. This means that it may not find the globally optimal subset of features that leads to the best overall model performance.

It may over-fit the data: If the number of features is significantly reduced, the model may end up fitting the training data too well, leading to overfitting and poor generalization to new data.

It does not consider interactions between features: Backward elimination only evaluates the importance of individual features and does not consider interactions between features. This means that it may eliminate features that are not important on their own, but are important in combination with other features.

Feature Extraction

- Linear combinations are particularly attractive because they are simpler to compute and analytically tractable.

- Given $x \in R^N$, find an N x K matrix U such that:

$$y = U^T x \in R^K \text{ where } K<N$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_N \end{bmatrix} \xrightarrow{U^T,\ f(\mathbf{x})} \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ y_K \end{bmatrix}$$

This is a projection from the N-dimensional space to a K-dimensional space.

Feature Extraction (cont'd)

From a mathematical point of view, finding an optimum mapping $\mathbf{y}=f(\mathbf{x})$ is equivalent to optimizing an **objective** function.

Different methods use different objective functions, e.g.,

- Information Loss: The goal is to represent the data as accurately as possible (i.e., no loss of information) in the lower-dimensional space.
- Discriminatory Information: The goal is to enhance the class-discriminatory information in the lower-dimensional space.

Feature Extraction (cont'd)

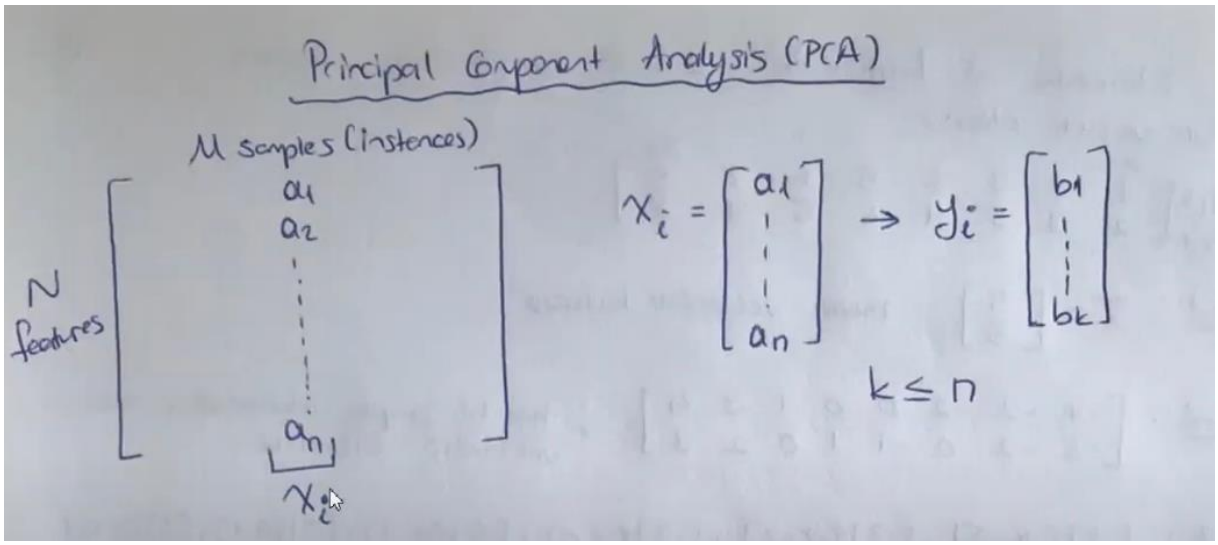
Commonly used linear feature extraction methods:

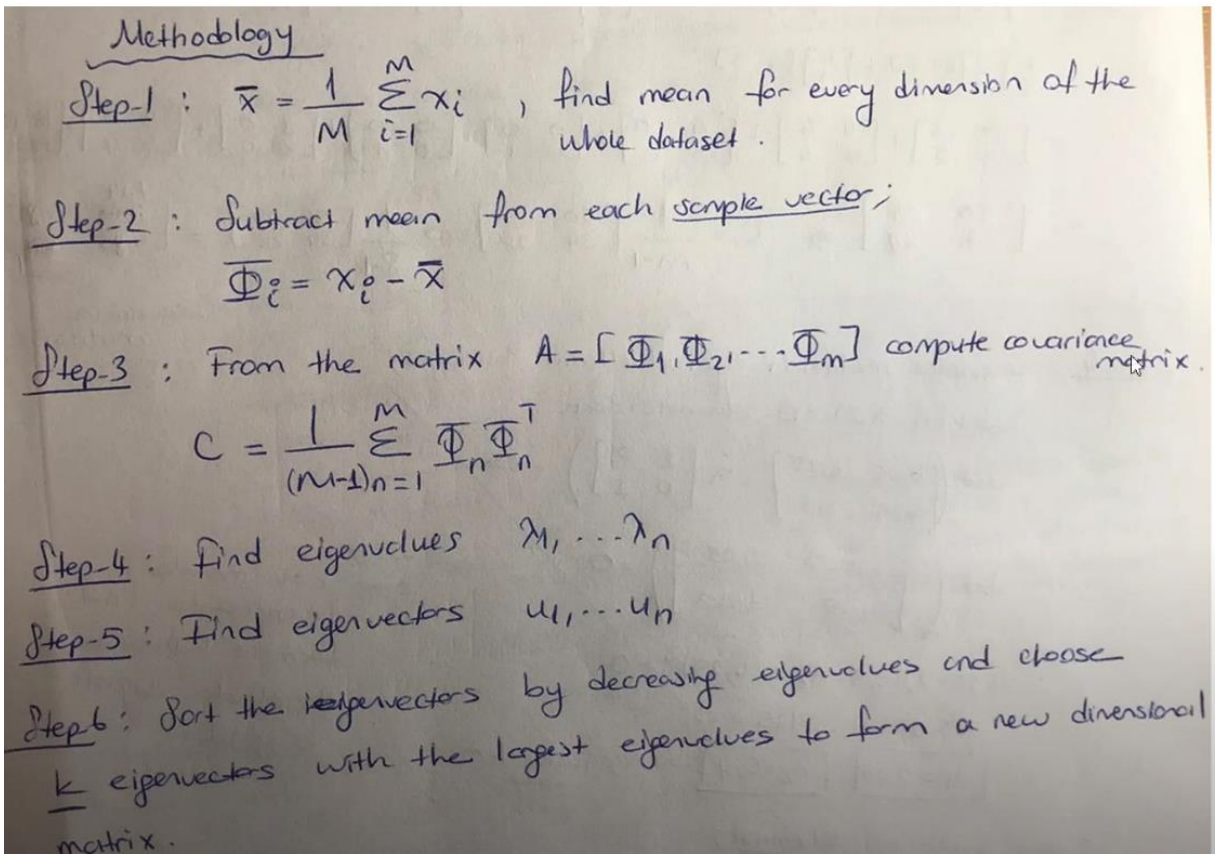
- Principal Components Analysis (PCA): Seeks a projection that **preserves** as much **information** in the data as possible.
- Linear Discriminant Analysis (LDA): Seeks a projection that **best discriminates** the data.

Some other interesting methods:

- Retaining interesting directions (Projection Pursuit),
- Making features as independent as possible (Independent Component Analysis or ICA),
- Embedding to lower dimensional manifolds (Isomap, Locally Linear Embedding or LLE).

PCA: Verideki en büyük variance değerini sabit tutarak işlem yapar. En büyük variance değerini eigenvectors kullanarak bulur. Eigenvalue'yü de kullanır.





PCA steps:

1- Bütün featureların ortalamaları (mean) bulunur.
2- Sample vectorden sırasıyla mean vectorler çıkarılır.
3- Yeni oluşan column matrix kullanılarak covariance matrix bulunur. (Kare matrix lazım olduğu için. N feature için NxN matrix elde edilir.)
4- Eigenvalues bulunur
5- Eigenvectors bulunur.
6- Eigenvectors azalan sırayla sıralanır. En yüksek eigenvalue'ya sahip K adet eigenvectors seçilir. (Yeni matrix için)

How do we choose K ?

- *K* is typically chosen based on how much information (variance) we want to preserve:

$$\frac{\sum_{i=1}^{K}\lambda_i}{\sum_{i=1}^{N}\lambda_i} > T \quad where\ T\ is\ a\ threshold\ (e.g., 0.9)$$

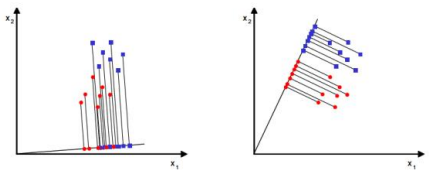
- If T=0.9, for example, we say that we "preserve" 90% of the information (variance) in the data.
- If K=N, then we "preserve" 100% of the information in the data (i.e., just a change of basis)

The main limitation of PCA is that it does not consider class separability since it does not take into account the class information.

- i.e., there is no guarantee that the directions of maximum variance will contain good features for discrimination.

Linear Discriminant Analysis (LDA)

- What is the goal of LDA?
  - Seeks to find directions along which the classes are best separated (i.e., increase discriminatory information).
  - It takes into consideration the scatter within-classes and between-classes.



Bad separability Good separability

LDA works by finding a linear combination of the original features that maximizes the separation between the classes, while also minimizing the within-class variance. The resulting combination is called the "linear discriminant," and the number of discriminants is equal to the number of classes minus one.

LDA can be performed using singular value decomposition (SVD) or eigenvalue decomposition of the scatter matrices of the data. After fitting the LDA model to the data, the transformed data can be obtained by projecting the data onto the linear discriminants.

LDA is a powerful method for dimensionality reduction, particularly for classification tasks. However, it requires that the data be normally distributed and that the classes have equal covariance matrices. It is also sensitive to the scaling of the data and may not be robust to outliers. Finally, LDA is not suitable for multiclass classification tasks, as it can only find a single linear discriminant.