# Machine Learning Review

Zhongjian Lin
University of Georgia

November 19, 2024

## Machine Learning

Machine learning is about extracting knowledge from data. There are input and output data. There are two types of learning: supervised learning and unsupervised learning.

- Machine learning algorithms that learn from input-output pairs are called supervised learning.
- In unsupervised learning, only the input data is known and there is no known output data given to the algorithm.

In supervised learning, we have two types of models: classification and regression.

- Classification model is used to analyse data when output is discrete/categorical.
- Regression model is for continuous output.

Measuring Success: Training and testing data

- For classification, we use accuracy of prediction as the measurement of model performance.
- For regression, we use $R^2$ as the measurement of model performance.

## Supervised Learning

- Nearest neighbors: for small datasets, good as a baseline, easy to explain.
- Linear models: Go-to as a first algorithm to try, good for very large datasets, good for very high-dimensional data.
- Decision trees: Very fast, don't need scaling of the data, can be visualized and easily explained.
- Random forests: Nearly always perform better than a single decision tree, very robust and powerful. Don't need scaling of data. Not good for very highdimensional sparse data.
- Gradient Boosted Decision Trees: Often slightly more accurate than random forest. Slower to train but faster to predict than random forest, and smaller in memory. Need more parameter tuning than random forest.
- Support Vector Machines: Powerful for medium-sized datasets of features with similar meaning. Needs scaling of data, sensitive to parameters.
- Neural Networks: Can build very complex models, in particular for large datasets. Sensitive to scaling of the data, and to the choice of parameters. Large models need a long time to train.

## Cross Validation

1. k-fold CV.
2. Stratified k-fold CV.
3. k-fold CV with random shuffle.
4. Grid Search with Cross Validation.

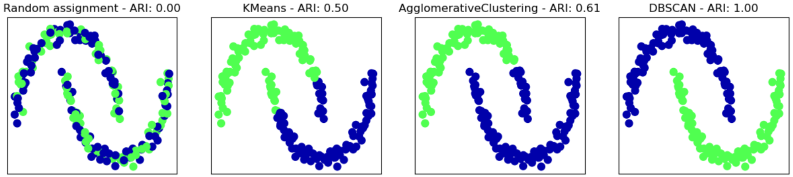
## Automatic Feature Selection

There are three basic strategies: Univariate statistics, model-based selection and iterative selection. All three of these methods are supervised methods, meaning they need the target for fitting the model.

- In univariate statistics, we compute whether there is a statistically significant relationship between each feature and the target. Then the features that are related with the highest confidence are selected. In the case of classification, this is also known as analysis of variance (ANOVA).
- Model based feature selection uses a supervised machine learning model to judge the importance of each feature, and keeps only the most important ones. The supervised model that is used for feature selection doesn't need to be the same model that is used for the final supervised modeling.
- In iterative feature selection, a series of models is built, with varying numbers of features.

## Unsupervised Learning

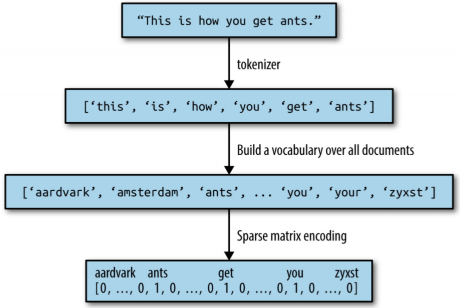
- Principal Component Analysis.
  - Principal component analysis (PCA) is a method that rotates the dataset in a way such that the rotated features are statistically uncorrelated. This rotation is often followed by selecting only a subset of the new features, according to how important they are for explaining the data.
  - We can use PCA for dimensionality reduction by retaining only some of the principal components.
- Clustering
  - k-Mean Clustering.
  - Hierarchical Clustering.
  - Density based spatial clustering of applications with noise (DBSCAN).



## Bag-of-Words

Computing the bag-of-word representation for a corpus of documents consists of the following three steps:

- Tokenization: Split each document into the words that appear in it (called tokens), for example by splitting them by whitespace and punctuation.
- Vocabulary building: Collect a vocabulary of all words that appear in any of the documents, and number them (say in alphabetical order).
- Encoding: For each document, count how often each of the words in the vocabulary appear in this document.


"This is how you get ants."
tokenizer
['this', 'is', 'how', 'you', 'get', 'ants']
Build a vocabulary over all documents
['aardvark', 'amsterdam', 'ants', ... 'you', 'your', 'zyxst']
Sparse matrix encoding
aardvark ants get you zyxst
[0, ..., 0, 1, 0, ..., 0, 1, 0, ..., 0, 1, 0, ..., 0]

## Word embeddings

One-hot encoding makes a feature-engineering decision. You're injecting into your model a fundamental assumption about the structure of your feature space. That assumption is that the different tokens you're encoding are all independent from each other: indeed, one-hot vectors are all orthogonal to one another. And in the case of words, that assumption is clearly wrong. Words form a structured space: they share information with each other. The words "movie" and "film" are interchangeable in most sentences, so the vector that represents "movie" should not be orthogonal to the vector that represents "film"—they should be the same vector, or close enough.

- The geometric relationship between two word vectors should reflect the semantic relationship between these words.
- For instance, in a reasonable word vector space, you would expect synonyms to be embedded into similar word vectors, and in general, you would expect the geometric distance (such as the cosine distance or L2 distance) between any two word vectors to relate to the "semantic distance" between the associated words. Words that mean different things should lie far away from each other, whereas related words should be closer.
- *Word embeddings* are vector representations of words that achieve exactly this: they map human language into a structured geometric space.

## The Transformer architecture

- A simple mechanism called "neural attention" could be used to build powerful sequence models that didn't feature any recurrent layers or convolution layers.
- The Transformer is effective for sequence data. We'll then leverage self-attention to create a Transformer encoder, one of the basic components of the Transformer architecture.

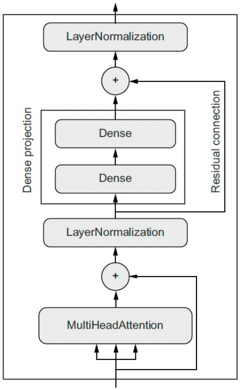


**Figure 11.9** The `TransformerEncoder` chains a `MultiHeadAttention` layer with a dense projection and adds normalization as well as residual connections.

# Language Model

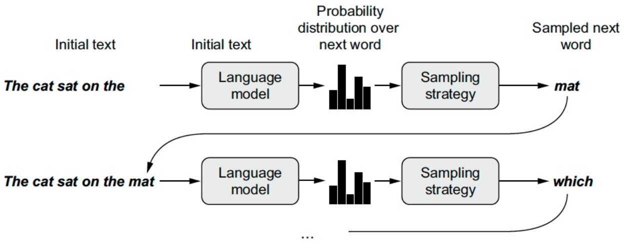


**Figure 12.1** The process of word-by-word text generation using a language model

## Sampling strategy

When generating text, the way you choose the next token is crucially important.

- A naive approach is greedy sampling, consisting of always choosing the most likely next character. But such an approach results in repetitive, predictable strings that don't look like coherent language.
- A more interesting approach makes slightly more surprising choices: it introduces randomness in the sampling process by sampling from the probability distribution for the next character. This is called stochastic sampling.
- In such a setup, if a word has probability 0.3 of being next in the sentence according to the model, you'll choose it 30% of the time. Note that greedy sampling can also be cast as sampling from a probability distribution: one where a certain word has probability 1 and all others have probability 0.
- Sampling probabilistically from the softmax output of the model is neat: it allows even unlikely words to be sampled some of the time, generating more interesting looking sentences and sometimes showing creativity by coming up with new, realistic sounding sentences that didn't occur in the training data.

## Neural style transfer

In addition to DeepDream, another major development in deep-learning-driven image modification is neural style transfer, introduced by Leon Gatys et al. in the summer of 2015. Neural style transfer consists of applying the style of a reference image to a target image while conserving the content of the target image.

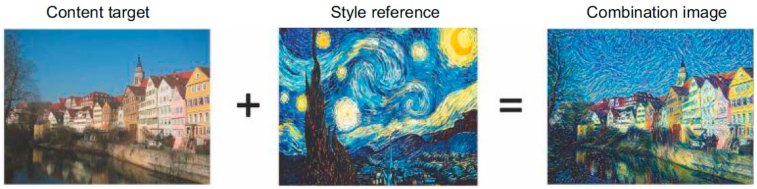


**Figure 12.9 A style transfer example**

## VAEs and GANs

There are two main techniques in this domain of image generation: variational autoencoders (VAEs) and generative adversarial networks (GANs). The key idea of image generation is to develop a low-dimensional *latent space* of representations. Once such a latent space has been learned, you can sample points from it, and, by mapping them back to image space, generate images that have never been seen before. These new images are the in-betweens of the training images.

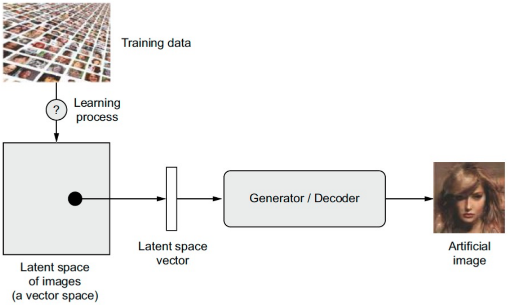
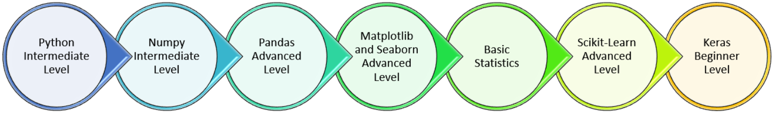


**Figure 12.13 Learning a latent vector space of images and using it to sample new images**


Python Intermediate Level
Numpy Intermediate Level
Pandas Advanced Level
Matplotlib and Seaborn Advanced Level
Basic Statistics
Scikit-Learn Advanced Level
Keras Beginner Level

enjoy the journey