01SU

Bc. Martin Kovanda
dle přednášek ...

10. října 2021

Obsah

Předmluva a poděkování

Doufám, že Vám tato příručka usnadní studium těchto předmětů a dostatečně Vás namotivuje k jejich absolvování u státnic. Tato příručka vynechává veškeré důkazy a v mnoha oblastech je zjednodušující. Je nicméně dělána tak, aby v ní bylo vysvětleno vše, co je potřeba k pochopení problematiky a k úspěšnému absolvování státnicového předmětu. Pro lepší pochopení předmětů je silně doporučeno navštívit přednášky.

1 introduction

Artificial intelligence is a system that can perceive the surrounding environment, evaluate it, make decisions, and execute actions to achieve a goal set by the user.
The perception is usually based on image data using specialized sensors. Evaluation of these data means recognizing objects that are contained in images, etc.
Decision making can be understood as an optimization problem under boundary conditions, e.g. for self-driving cars we know they don't fly, etc.
There is a **general** artificial intelligence, which (for now) belongs to the science fiction and it is an intelligence capable of everything that a human or other intelligent being can do. Nowadays they are working with **specific** artificial intelligence capable of solving only specific problems.
Machine learning can be divided into **classical methods** (handcrafted features) and **deep learning** and other methods. Classical methods generally require less data and require the human to choose some parameters. On the other hand, deep learning needs a large amount of data but then finds the parameters itself. The second problem is that neural networks do not provide an explanation for why they give the results they do.

Definice 1.1 (Pattern Recognition (PR))**.** Assigning a pattern/object to one of pre-defined classes.

Definice 1.2 (Statistical (feature-based) PR)**.** The pattern is described by features, i.e. an n-D vector.

There is a **Supervised PR** and **Unsupervised PR**. The supervised has training set available for each class. The Unsupervised PR sometimes does not even have the number of classes set.

Desirable properties of the training set

A good dataset contains typical representatives of each class including intra-class variations. It should also be reliable and large enough. If possible, the data should be annotated by domain experts. There will always problems in the training data, such as expert mistakes, wrong annotations, misunderstanding data etc.
There are several desirable properties of the features. (!!! missing text Lukáš) **Invariance**, **Discriminability**, **Robustness**, **Efficiency, independence, completeness, etc.**

Definice 1.3 (Complete feature space (úplný příznakový prostor))**.** Is a feature space from which the original image (or another data) can be fully reconstructed.

Definice 1.4 (Independence)**.** Independence means that any feature cannot be made from other features.

In order to make a good recognition model, it is necessary to make all the objects from the same class close to each other in the feature space, while data from different classes far from each other. As an example, letters "C"and "G"have a curve in common but in the feature space they must be far. Another crucial thing is to prevent overfitting, i.e. making a classifier 100% perfect at the training data. This classifier can be completely useless on new data.

Each class is characterized by its discriminant function $g(x)$. Classification is then a maximization of $g(x)$. Asigning x to class i if

$$g_i(x) > g_j(x), \qquad \forall j \neq i.$$

Discriminant functions define decision boundaries in the feature space.
In the process of classification we try to maximize $\max_i p(w_i|x)$ given a data x. We then choose a class which has the highest probability.

Příklad 1.5. If we have a 2D feature space with 2 classes with Gauss probability

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\Big[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\Big],$$

the resulting classes borders can be any conic section, e.g. a line, parabola, hyperbola etc. Note that in the case of hyperbolic borders there will be actually 2 curves.

!!! Lukáš add eigenvectors etc.

Poznámka 1.6. The classificator is linear ($G_1(x) = G_2(x)$ is a line) if the covariance matrices of both Gauss distributions are identical. This fact can be used to simplify the calculation, however, this usually is not an issue. On the other hand, if we assume the classifier is supposed to be linear, it can be used to calculate the covariance matrix from all the data at only (and not per category as it would need to be). The linearity assumption is often called forced linearity. This is possible since the covariance matrix only depends on the variance and not the position. In case of $p_1 G_1(x) = p_2 G_2(x)$, the shape of the decision borders do not change but instead they shift towards the Gauss with a smaller p.

Poznámka 1.7. Note that when the feature space has n dimensions, the covariance matrix has n^2 values.

Poznámka 1.8. The Bayesian classifier is suitable if enough data for the distribution estimate is available. Such areas of application might be for example e-mail filtering or image processing (multispectral remote sensing).

Definice 1.9 (Mahalanobis distance)**.** We define **Mahalanobis distance** as

$$(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}).$$

1.1 Minimum distance (NN) classifier

The Minimum distance classifier is the simpliest classifier. It is based on a pre-defined distance function $d(\mathbf{x}, \omega_i)$ and classify a new item based on the closest item present in the train set. Generally, NN classifiers are not linear. The basic NN classifier is sensitive to outliers and hence is usually not good for predictions. The resulting decision border is a broken line.
An alternative is to take only the center of all the classes and calculate distance from these centers. In this approach we throw up the information about variance of particular classes.

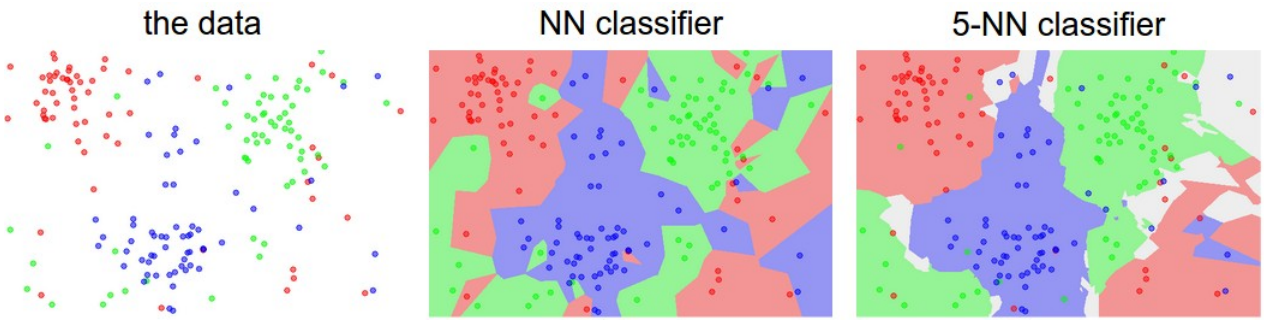
1.1.1 k-NN classifier

There are two versions of this algorithm. In the old worse one we calculate k nearest neighbours and choose the class which is most present in those neighbours. However, many times there is the same amount on neighbours present in different classes and we would have to define what to do next. In the upgraded version k we calculate nearest neighbours until some class reach k neighbours.

This method is good when there are outliers in data, because k-NN classifier is robust to (k-1) outliers. The problem occurs when there are more outliers next to each other. Another problem of k-NN is a huge computational demand. Therefore there are many algorithms which try to surpass this problem somehow.

Poznámka 1.10. Note that 1-NN classifier is NN classifier.

Poznámka 1.11. Usually k is a small number.



Obrázek 1.1: The difference between NN and k-NN classiffier.

1.2 Simple training algorithms

1.2.1 Perceptron

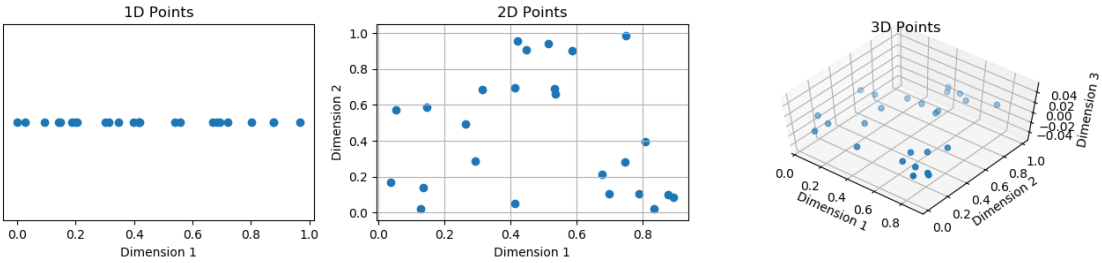
One of the simplest "neural network"is called a Percepton. https://en.wikipedia.org/wiki/Perceptron (.... chtělo by dodělat, pokud někdo chce ....).

1.2.2 SVM - Support Vector Machine

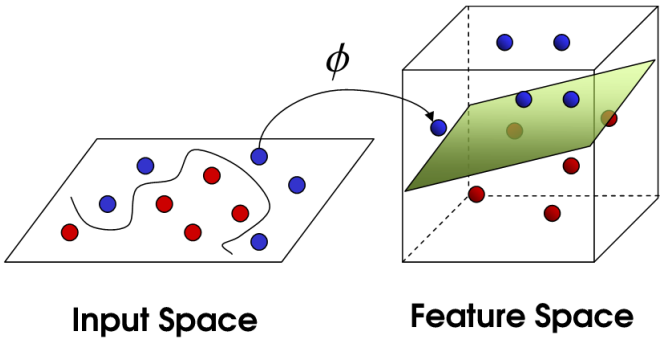
The idea behind SVM is to solve some problems occuring in the simple classifiers. SVM creates 2 parralel separating lines (the decision borders) which would be the furthest possible from each other. The resulting classifier then uses the middle line between these 2 created lines. It is an "optimal"linear classifier, which focuses on maximizing the margin.
But SVM has one fatal problem. All the line prediction depends only on some of the border items. Many times the class border data are kind of a noise and these data are usually outliers. A possible solution to this problem is to allow some mistakes (training data which will be classified wrong). A version which uses this is called the **Soft Margin SVM**. This method requires a manual setting of the allowing coefficient.

Poznámka 1.12. Any classifier **should not** depend od extremal values in practice!

SVM only works on linear separable data. A way how to solve non-separable data is to enlarge the dimensionality (increasing the number of features) and make them separable. There is however a **Curse of dimensionality** which says that the data soon become very sparse and a good classifier would require a lot of training data. Another option is to map the features into another space ("**The Kernel trick**"), but this might lead to over/undertraining of the model.



Obrázek 1.2: Curse of Dimensionality.



Obrázek 1.3: The Kernel trick.

1.2.3 Classifier performance

How to evaluate the performance of the classifiers? There are two possibilities:

- evaluation on the training set (**optimistic error estimate**) - not recommended
- evaluation on the test set (**pessimistic error estimate**)

In order to evaluate the performance of the classifier is to evaluate it on test set. Test dataset contains data on which the classifier did not train itself and which were chosen in the same way as the training dataset. After this we can for example calculate the **confusion matrix** or other statistics, such as accuracy, precision, recall etc.

		CLASSIFIER				
		1	2	3	4	...
GROUND TRUE	1	20	0	0	3	...
	2	7	8	6	0	...
	3	1	0	15	5	...
	4	6	0	7	23	...
	...	⋮	⋮	⋮	⋮	...

Obrázek 1.4: Example of a Confusion matrix.

Poznámka 1.13. Perfect classifier would have diagonal confusion matrix. Be careful while interpreting the results given by confusion matrix - the classification classes might not be uniformly distributed and absolute numbers might be misleading. Also, in many cases (typically in medical environment), not all mistakes are considered equal and we might focus on reducing the most several ones. For example while diagnosing patient with an illness, it is desirable to have "false negative"error as low as possible, while "false positive"is usually not such a bother.

One way how to choose the training and test dataset is to use the cross-validation where we choose the training and test datasets randomly and calculate particular confusion matrices. The final step is to train the classfier on all the data (and believe that it will be slightly better than all those cross-validated sets). In case that the confusion matrices are very different, something is wrong with the classifier or the data.

2 Unsupervised Classification (cluster analysis)

Clusters are hardly defined as a compact well-separated subsets of data. There is no good formal definition. One of them is e.g. that a cluster is a subset in which it is possible to jump to another item with a step smaller than some t. However, even a small change of t can significantly affect the subsets. For this reason, no definition is used and in practise any partition of the data is considered as a cluster. Instead, algorithms are compared by how good clusters are they making.

Definice 2.1 (Ward criterion)**.** In order to compare clustering, we define Ward criterion for N Clusters as

$$J = \sum_{i=1}^{N} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \mathbf{m}_i\|^2 ,$$

where $\mathbf{m}_i$ is a center of mass from the i-th cluster and C_i is a set of items belonging to the i-th cluster.

In real problems the variance measure J from the Ward criterion should be minimized (but not with N since $\min_N J(N)$ is monotone descreasing). There are some problems with this criterion, such as when there is a small cluster and a big one, it tries to make two similar size clusters.
When N is known, we can use the **Iterative methods**. When the N is unknown, we can use the **Hierarchical methods**. There are also algorithms combining these two groups.

2.0.1 Iterative methods

N-means clustering

In N-means clustering we choose N initial cluster centroids, then we classify every point x according to the minimal distance. From this we get N clusters. In each of them the center of mass is computed and taken to the next iteration. The iteration ends when the centers do not move. In other words, the algorithm is following

1. select N initial cluster centeroids;
2. classify every point $\mathbf{x}$ according to minimum distance;
3. recalculate the cluster centeroids;
4. if centeroids did not change, then STOP, else GO TO 2.

There are many drawbacks such as the results highly depend on the initial center choice and J is not minimized. Also, the results are often intuitively wrong. For this reason, it is generally a good idea to run all the process many times with different initializations. On the other hand, compared to other methods this algorithm is very fast and simple.

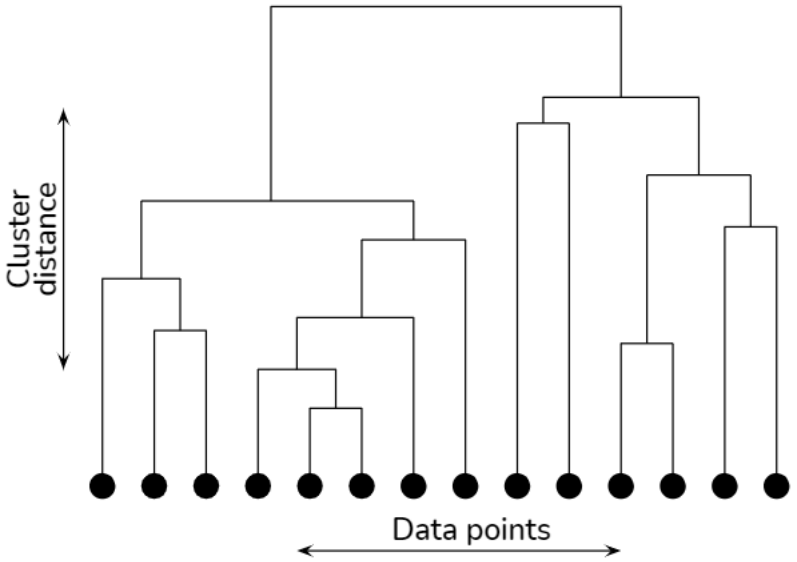
2.0.2 Hierarchical clustering methods

In Hierarchical clustering, we distinguish between two approaches. In the first one, **Agglomerative clustering**, we begin with as many clusters as there are items. In each step two

most similar (closest) clusters are joined together. One option is to calculate the Ward criterion for every possible change and choose the one that increases the Ward criterion with a minimal possible value. On the other hand, we can use center of masses of the maximum or minimum from the items already present in the cluster. The iteration ends when a strongly defined stopping rule is met or when there is too big increase of the Ward criterion compared to the previous ones. The result can be show in a clustering tree called the **dendrogram**. Algorithm summary:

1. each point = one cluster;
2. find two "nearest"or "most similar"clusters and merge them together;
3. repeat 2. until the stop rule is reached.

The second approach is called **Divisive clustering**. The difference is, that we begin with one cluster and in each step we split the cluster(s) into two smaller ones.



Obrázek 2.1: Example of dendrogram. As a measure for cluster distance Ward statistics J can be used.