Machine Learning for Urban Computing

Bilgeçağ Aydoğdu, Albert Ali Salah[1]

1. Introduction

Increasing urbanization rates and growth of population in urban areas have brought complex problems to be solved in various fields such as urban planning, energy, health, safety, and transportation. Urban computing (UC) is the name of the process that involves collection, integration and analysis of heterogeneous urban data coming from various types of sensors in the city, with the purpose of tackling urban problems (Zheng et al., 2014). Within this framework, the first step of UC is to collect data through sensors like mobile phones, surveillance cameras, vehicles with sensors, or social media applications. The second step is to store, clean, and index the spatiotemporal data through different data management techniques by preparing it for data analytics. The third step is to apply different analysis approaches to solve urban tasks related to different problems.

Machine learning methods are some of the most powerful analysis approaches for this purpose, because they can help model arbitrarily complex relationships between measurements and target variables, and they can adapt to dynamically changing environments, which is very important in UC, as cities are constantly in motion. Machine learning (ML) is a subfield of artificial intelligence (AI) that combines computer science and statistics with the objective of optimizing a performance criterion by learning models from data or from past experience (Alpaydın, 2020). The capabilities of ML have sharply increased in the last decades due to increased abilities of computers to store and process large volumes of data, as well as due to theoretical advances.

Especially in domains where human expertise is unable to crystallise the path between data sources and the performance measures, ML approaches offer new ways of model creation. In UC, data sources are rich and heterogenous. From a ML perspective, using heterogeneous sources of data as input to models is challenging, as most of the existing paradigms are designed to process one type of data. Additionally, urban data acquisition brings its own challenges. Urban data may come from sensors equipped by infrastructure elements, from public transportation systems, from satellites, from household surveys, governmental institutions, and even from people who act as sensors (called *citizen science*), either pro-actively supplying information by reporting issues, or by using mobile phones or applications whose data can be read to infer many things about the city and the way it is used by the people. These data sources have different bias and variance characteristics; each sensor comes with its own data gaps (e.g. mobile phone usage may exclude small kids, satellite coverage cannot include what goes on inside large buildings, public transport figures ignore other means of transport, etc.), and each data source is sampled at a different rate. These are all challenges in building unifying models.

Apart from detecting patterns in urban usage, ML methods can be used for detecting outliers, such as anomalies in the daily life of the city. For example, mobile phone activity at a certain location can be monitored for automatically detecting unusual patterns (Gündoğdu et al., 2017). Combining ML with computer vision allows us to detect problems in urban settings visually, and to classify areas of the city in terms of safety, wealth, accessibility, etc. (Santani et al., 2015). Another important usage is in prediction of urban activity based on past data, such as predicting queues in gas stations, which can be used for improving waiting times across the city (Zheng et al., 2013). ML can also help with knowledge extraction, providing interpretable insights into data collected from urban sensing.

---

[1] The authors are with Dept. of Information and Computing Sciences, Utrecht University. This is the uncorrected author proof. Please cite as: Aydoğdu, B., A.A. Salah, "Machine Learning for Urban Computing", in S. Carta (ed.), *Machine Learning, Artificial Intelligence and Urban Assemblages: Applications in architecture and urban design*, Wiley, forthcoming.

Furthermore, novel *explainable ML* approaches are providing more insightful models that help clarify their predictions to human analysts.

The improvements in urban sensing technologies, increased pervasiveness of these technologies and better data management techniques all contribute to improving analysis. It is however important to understand what the different ML methods are, what they can offer, their strengths and weaknesses, as well as their applications to UC, to be able to effectively choose the most suitable method (or sometimes to decide not to use any ML method) for solving the relevant problems in the urban setting.

In this chapter, we discuss the key techniques in ML and their use in various UC scenarios. Firstly, we introduce the main concepts and key techniques of ML, followed by relevant issues while designing any ML study. This is a rapidly growing area of research, and we will not attempt to cover all approaches, nor will we provide a mathematical exposition. However, we will point the reader towards resources for following up the main points of discussion in this chapter.

2. Key terms in machine learning

One of the most important steps in ML is the abstraction of the problem, which is the link from sensing (or raw signals from the world) to the tasks designated to be solved with ML. In UC, multiple data sources can be tapped for feeding the ML algorithms, such as public infrastructure usage, camera feeds, social media data, mobile phone data, satellite images, etc. When we look at the raw signals, they will come in different sizes, resolutions, and frequency. The first step in the ML pipeline is to create meaningful **features** from these signals. In creating features, domain expertise may play a great role. Take for example, an application that monitors train crossings via camera to learn anomalous pedestrian behaviours to prevent suicides. In such an application, a machine learning based system is used to filter potentially suspicious situations, and a human operator will evaluate the filtered results. But the raw camera signal may be too coarse and high-dimensional to process directly with the ML model. If we can detect pedestrians near the tracks, and if we can encode their movement trajectories by the movement of a single point, we obtain a much sparser and more informative representation. This representation loses some information; if the clothing of the person is informative (e.g. wearing black as a possible indication of depressive mood), then the point representation will not encode this information. However, the **dimensionality** of the feature is much reduced, compared to the full video signal.

The **curse of dimensionality** refers to the phenomenon that most ML approaches scale poorly with the dimensionality of the problem: they require an increasing amount of samples to learn the problem structure properly. This increase is super-linear, and modern deep learning approaches that have millions of free parameters need millions of samples to find proper values for these parameters. Subsequently, it is essential to find a good set of features.

Two key approaches to obtaining features are feature extraction and feature selection. In **feature extraction**, the original signals are transformed into more informative or sparser feature sets. It is also possible to see multiple feature extraction steps in a modern ML pipeline. In **feature selection**, on the other hand, the original features are preserved, and a subset is selected for use. Since original features could be strongly correlated (e.g. adjacent pixels in a street camera image), adding one feature when the other is already included is not too informative. Feature selection approaches try to select features that are complementary, by reducing redundancy (see, for example, Peng et al., 2005).

Feature extraction methods come in all flavors. A signal processing approach, for example, is to use a theoretically complete basis (such as a **wavelet**, or a **Fourier transform**), which will encode the raw signal in a coarse-to-fine resolution, and then discard most of the finer parts to get a feature of

desired dimensionality. But increasingly, we see that ML methods developed for one task to be used as feature extractors for another task. A good example is a deep neural network trained on a broad object recognition task, repurposed for a related but different task (Chatfield et al., 2014). One can conceptualize a (deep) neural network as a series of operations performed on an input feature vector (such as a camera image). These operations are multiplications with matrices computed during the **training** of the network, non-linear transformations, convolution operations to detect the presence of features, and pooling operations to provide some translation invariance to the detected features. When repurposed for a different task, the ML approach is treated as a black box that extracts good visual abstractions in order to do its job, and by peeling off the outer layer of the network, we can take some of its internal representations as sparse and powerful features for a different task. This approach falls under the rubric of **transfer learning**, and the more similar the tasks are, the more powerful the features will be. This approach is particularly useful if we have few samples to learn the original task properly, but many samples for learning the transferred ML system, or if the transferred system is already developed and trained elsewhere. It is possible to obtain many such trained models from online repositories like GitHub (https://github.com/), or Model Zoo (https://modelzoo.co/).

Machine learning approaches can be classified into several broad categories, depending on the assumptions they make. A lot of UC tasks are either classification or regression problems. Given a set of classes and a set of input instances, in which each instance belongs to one or more classes, the goal of a **classification** model is to learn the boundaries that **discriminate** instances of each class from instances of all of other classes. In the **training phase** of the ML model, a family of models is explored for the task, and a **training set** of instances are used to find the most suitable model, as well as the best parameter values for the model. For example, suppose we want to classify gas stations in a city as "busy" or "idle", based on camera images. This is a two-class classification problem where the training data will look like $X = \{I^t, r^t\}$, with $t = 1...N$, where I^t are camera images, r^t are the **labels** denoting busy or idle stations, and N stands for the total number of training images. If we are provided with the labels during training, this is called a **supervised learning** task. If, on the other hand, we are just provided the images, and we need to also discover how many classes we want to use in the classification task, this would be an **unsupervised learning** task, which is naturally harder. Typically, supervised learning tasks are concerned with finding discriminating boundaries, whereas unsupervised learning tasks are concerned with modeling the distributions of features, in tasks like **clustering**, which is finding groups and assigning data points to these groups, and **density estimation**, which is fitting a statistical distribution to the data. If the labels are drawn from a continuous set (e.g. a busyness index for the gas station that can take values between 0 and 1), then we deal with a **regression** problem. An example of a regression problem in UC is next location prediction for observed vehicles, which can be used for optimizing traffic flows and predicting traffic jams (Barlacchi, 2019).

When we measure the performance of the ML model in a supervised classification task, we use a **test set** of instances which are not used during learning. This is how the ML algorithm is tested for checking whether it has "memorised" the training samples, in which case it will have poor generalization to novel samples. The most important pitfall in ML is to train multiple models on the same dataset and pick the best performing model to report the performance. This is called **cherry picking**, and it is not unlike doing a lot of statistical tests without adjusting significance levels for multiple tests. We expect to see performance reported on an independent test set, which is not used too frequently to query performance. If the test set is used in model selection, it actually becomes a part of the training set. Consequently, a proper test protocol is one of the first things one needs to check when looking at an ML paper critically. Many approaches use an additional **validation set** to fine tune the parameters of ML models, and **cross-validation**, where the training data is separated into

multiple folds that serve as training and validation sets, alternatively. It is important to have the test set separated from the cross-validation set for an independent evaluation.

The second important check in evaluating an ML approach is the quality of the labels. The error of an ML model is typically formulated with respect to some **ground truth**. Labeling data usually requires humans or a group of experts to annotate the data, and hence, it is costly, and depending on the application, sometimes subjective. However, we see more and more clever approaches in obtaining automatic labels. For example, in an application where the Government monitors the citizens' mood in a city using geo-tagged Tweets, the emojis in the Tweets could be used as providing sentiment ground truth for the text (Li et al., 2016). Then, an ML approach can be trained on the Tweets with such ground truth, and then used to predict the sentiments on the ones that do not have emojis. It is also possible to let one reliable modality supervise a less reliable one. If we have a few air quality measuring sensors deployed in the city, we can use the ground truth they provide to train a satellite image based, or a social media based system. Finally, **citizen science** allows the inhabitants of a city to provide labels and training samples, because the inhabitants are incentivized to do that if ultimately it helps the life in the city (Kobori et al., 2016). For example, they can share their location when using a map application which gives recommendations based on traffic, as the collected location information will improve ML based traffic predictions, and thus improve the service they are receiving.

An important concern is the **bias-variance dilemma** for ML models. **Bias** refers to the tendency of a system to prefer certain outcomes over others, thus it explains how an ML model generalizes to new samples (Mitchell, 1980). **Variance,** on the other hand, is how much the models can change with changes in the training data. Complex models with many parameters will have less bias as they are flexible in fitting the training data, but more variance, when compared to simpler models. Ideally, the model complexity should be “just right” for a given problem. This is often easier said than done. Subsequently, there are many approaches for selecting the correct model complexity, and expert knowledge can also help a lot. In the context of UC, machine bias can be potentially very harmful, for example if the ML system is trained to prefer outcomes aligned with certain subpopulations in the city. It is not always straightforward to test these systems for bias. For example, we can test whether a system that works on data from individuals performs differently for male and females, for kids and adults, or for individuals from different ethnic backgrounds. But the bias may be more complex, and may only be detected by looking at intersections of groups. It can be rooted in unbalanced training data, but it can also arise from the algorithmic details (Mehrabi et al., 2019).

3. Key methods in machine learning

3.1. Classification

There are many examples of supervised ML problems in UC. An example application of **classification** algorithms in UC is to infer different land usages of a region based on observed spatial interactions coming from various big data sources (Liu et al., 2016; Toole et al., 2012). Here, each point on the map of the city is assigned to one of a small set of classes, for example ‘residential’, ‘commercial’, ‘industrial’, ‘park’, and ‘other’. Depending on the data source, ML techniques usable for this problem will show differences. In (Toole et al., 2012), aggregated mobile phone data is used as the input. This type of data can be obtained from telecommunication companies under strict privacy regulations, and can be very informative for mobility-related analysis (Salah et al., 2019). For each location, the phone usage of people in that area will increase and decrease during different hours of the day, and it is this dynamic that will be leveraged for the classification. Increased use during work hours will show that the spot is commercial or industrial, whereas increased use on the

weekends and on particularly sunny days will suggest that it's a park. Using the **ground truth** labels of a few locations will enable supervised approaches.

Two-class problems are the simplest classification problems, where the aim is to find a **discriminating boundary** (also called a **discriminant**) between the classes in some feature space. Linear discriminant analysis (LDA) is a classic method that tries to find a **linear discriminant,** which maximizes the variance between the two classes it separates, and minimizes the variance within each class. The **support vector machine** (SVM) is also a supervised learning algorithm used for essentially finding a discriminating boundary between classes. While LDA works by projecting the data on a subspace, SVM views this as an optimization problem, and tries to find the optimal hyperplane that separates each class (Cortes and Vapnik, 1995). The data points belonging to the two classes may not be separable by a line in the original feature space. SVM uses a **kernel** to project the points to a higher dimensional space where they become linearly separable. The distance from the discriminating boundary to the closest samples of the two classes is called the **margin,** and SVM tries to maximize the margin. The larger the distance between the projected classes, the more difficult it will be to make a mistake in classification. SVM is a popular method in a number of UC tasks such as urban logistic forecasting (Gao and Feng, 2009), road traffic prediction (Saldana-Perez et al., 2019), and the classification of satellite images based on different urban objects they include (Giriraja et al., 2014).

3.2. Artificial Neural Networks

Artificial neural networks (ANN) (McCulloch and Pitts, 1943) are cognitively inspired models and they have a lot of applications in ML. The multilayer perceptron (MLP), is a non-parametric estimator that is used for classification and regression. The term **non-parametric** in this context refers to the fact that these models do not have distributional assumptions about the input.

In its most basic form, a simple set of equations govern the computation in the MLP. If x denotes the d-dimensional input (typically enhanced with one fixed dimension called "bias term" which always has the value of 1, used for shifting the fit on the data) the output y of the MLP is computed as:

$$z = \delta(w^T x),$$

$$y = v^T z.$$

Here, w is a (h x d)-dimensional matrix of coefficients, which projects the d-dimensional input to a h-dimensional feature space (called the **hidden layer**, as this is an internal representation), δdenotes a non-linear activation function that increases the representation power of this projection (typically sigmoid, tangent hyperbolic, or rectified linear functions are used), and v is a second set of coefficients that project the intermediate z values (i.e. hidden unit outputs) to the output feature space. In a regression problem, this setup can be used to define a regression function as $f(x) = y$. In a classification problem, typically a **one-hot encoding** is used, where for C classes, a C-dimensional vector is the output, and the class membership is indicated by setting the correct class or classes to 1, and the rest to 0. For example, in a three-class problem, the desired 3-dimensional outputs will be 100, 010, and 001, denoting membership of each of the three classes. The output of the MLP will be interpreted by looking at which of these class codes is the closest to the actual output. The first part of the model (expressed by z) is a **perceptron** (Rosenblatt, 1961), and the second part is what makes the model multi-layer.

The addition of more layers to such a **feed-forward** model results in a **deep neural network**. These models have layers with different processing capabilities. The most important of these are **convolution layers**, where a small, local portion of the input (or an intermediate representation) is multiplied with a small matrix of coefficients, which acts like a localized feature detector. The

convolution layer seeks for these features over the entire input. The **pooling layers** collect values in a small neighbourhood and perform a selection, for example by recording the maximum value. This type of layer provides location invariance; if a feature is detected in different parts of the input, the pooling layer is used to propagate a fixed representation for the presence of this feature. This is obviously very valuable in image-based problems, where the input is the entire image, and features (imagine detecting people or cars) can be at different locations of the image. **Convolutional neural networks** (CNN) are a type of deep neural network models that contain such convolution and pooling layers.

The **training** of MLP models, i.e. the setting of the weight matrices w and v for a given problem, is performed by formulating the error of the model on the training set, and by finding the weight values that minimize this error. For this, the true output values are required for each input, which makes MLP a supervised approach. Since the formulation cannot be analytically solved, neural network approaches are typically trained with iterative procedures, such as **backpropagation** (Rumelhart et al., 1986). In deep neural networks (also called **deep learning**, DL), backpropagation is not effective, and **stochastic gradient descent** is typically used. In both approaches, the gradient in question is the gradient of the error term, and we try to move the model along the gradient that reduces the said error.

Deep neural networks , as well as MLP, can be used for both supervised and unsupervised learning. In the latter, since we lack labels, the input itself, or its transposed version is used as the output. In order to ensure that the input is not simply copied to the output, these **self-supervised** approaches keep the dimensionality of the intermediate representations lower than the original dimensionality of the problem. This way, the model is forced to generate good **latent representations** for the problem. These low dimensional features are supposed to adequately represent the variance in the high-dimensional, original problem. In UC, DL is successfully applied to mining spatio-temporal features from multiple data sources for modelling regions of interest (Jiang et al., 2018), for identifying communities (Ferreira et al., 2020), as well as to learn feature representation from multiple big data sources (Liu et al., 2020).

3.3. Pattern discovery and clustering

Unsupervised learning methods are employed to find regularities, groupings and patterns in the input data. Common tasks include reducing dimensionality and clustering, which are effective to discover new insights from data. **Principal component analysis** (**PCA**) and **factor analysis (FA)** are the most frequently used approaches in dimensionality reduction and feature extraction. These methods make different assumptions, but each one is at its core a projection of the data to a different subspace, where the patterns will become more distinct. PCA selects the projection that maximizes the variance of the data, and by discarding low-variance dimensions, dimensionality is reduced (Jollife and Cadima, 2016). FA assumes that high-dimensional input variables are generated from a low-dimensional, latent space (Everett, 2013), and produces similar results with PCA. In UC, unsupervised learning is used in problems such as learning an embedding space for regions (Jenkins et al., 2019), location recommendation (Yang et al., 2018) or inferring whether people in a city are visiting or locals by their visitation patterns (Huang et al., 2017).

An important application of unsupervised ML approaches is **density estimation**. If we assume that the distribution from which the samples are drawn is known up to a number of parameters, we can apply **parametric** methods in ML, and try to find the best values of these parameters from the training set. Conversely, if assuming that the sample comes from a single known distribution yields large errors, a category of more flexible approaches can be useful.

Non-parametric ML approaches do not assume a model over the input space, but try to map the input data to the target values based on similarity. Another name for non-parametric approaches is

instance-based learning, because for each test instance they compare it to previous memorized training instances and interpolate an output value as a result. As an example, **decision trees** (Quinlan, 1986; Quinlan, 1990) are non-parametric models used in supervised learning tasks like classification and regression. They use a divide and conquer strategy to divide the input space into regions of decisions, through which each training instance is mapped to a label in classification, or to a numerical value in regression. The results of such models are usually highly interpretable, because the model looks at a number of features one by one in order to produce its decision. For a specific output, we can trace these decisions in the model to see how the output was generated.

There are also **semiparametric** approaches, which assume that there are different groups, or clusters in the data with known distribution models. The models include commonly used unsupervised learning algorithms mentioned above, which are also popular in UC problems like discovering the function of the urban land, segmenting the districts based on land use, or identifying the community structures (Frias-Martinez and Frias-Martinez, 2014; Yuan et al., 2012, Ferreira et al., 2020). The goal in semiparametric approaches is to learn the group densities and their proportions, given the sample and the number of assumed clusters. A **mixture model** describes the input density as a mixture of probability distributions, where each data point comes from one of these distributions, and there is a **mixing probability** to determine the relative weight of each of the mixture components. The most frequently used of such models is a **mixture of Gaussians,** where each component has a Gaussian distribution, specified by a mean vector and a covariance matrix.

An important problem in unsupervised density estimation is to find the parameters of components, as well as assigning which sample belongs to which component. **Expectation Maximization** (EM) algorithms (Dempster et al., 1977; Redner and Walker, 1984) solve these problems jointly. **Clustering** problems are an important class of problems in UC, where the input is assigned to one of several clusters of data points that are similar to each other in some sense. For example, the population in the city can be clustered (or grouped) according to their income, and depending on the resolution of the analysis, different numbers of clusters can be used. The popular ***k*-means** clustering algorithm starts by picking up k samples from the data, and computes distances of all points to these k samples. Each sample is assigned to these k centers (also called "means"), and then the center locations are updated by re-computing the mean value of all assigned data points. This procedure is repeated until convergence, and the data will be represented by k clusters.

This algorithm is in fact a special case of the EM algorithm used for fitting a Gaussian mixture model to data, but it uses a Euclidean distance between points, and treats all dimensions equally, as opposed to using a Mahalanobis distance, which will pay attention to the variances and covariances to compute distances between points in the input space.

The mixture models cluster the data by assuming that they are coming from known distribution models. Another way of clustering the data is without making such assumptions is **hierarchical clustering**. Here, the entire dataset will be broken down into groups hierarchically, by looking at distances between data points. It is also possible to start from points and proceed with grouping. The **agglomerative clustering algorithm** starts from many groups that contain only one training instance and proceeds by joining the closest groups until there is one group left. The desired solution is selected from an intermediate cluster assignment.

3.4. Bayesian approaches

When treating observations probabilistically, we may have some prior information that helps structure our uncertainty in the observation process. **Bayesian approaches** aim modeling our state of knowledge, where prior knowledge is suitably modified by empirical evidence, which arrives in the form of observations. In a city, we may for example know where the residential, work, and entertainment areas are, but a city is always changing, and new observations can modify our prior knowledge. Bayesian approaches are especially useful in modeling dynamic systems.

At the heart of these approaches, we find the Bayes rule:

$$p(\theta|X) = \frac{p(\theta)p(X|\theta)}{p(X)}$$

,

in which a set of observations X is related to a model represented by a set of parameters Θ. $p(\Theta)$and $p(X)$are the prior probability of the model and the observations, respectively, whereas $p(X|\Theta)$is the conditional probability of observing X under these parameters (called the model likelihood), and $p(\Theta|X)$ is the conditional probability of a certain set of parameters, given the observations (called the posterior). Using this formulation leads to many ML approaches that use evidence to train and fine-tune model parameters, and such models typically give probabilistic outputs, which can be interpreted as a confidence in the model outcomes. Model search is performed by **maximum likelihood** estimation and **maximum a-posteriori** estimation, which practically seek the best alignment of the data distribution and the distribution posited by the model.

There are many probabilistic approaches in ML (Ghahramani, 2015). The **naive Bayes** classifier is one of the simplest methods, where the features of the input are assumed to be mutually independent, given the class model. The maximum a-posteriori (MAP) decision rule classifies an input into the most probable class by just multiplying the conditional probabilities for each feature. **Graphical models** (Koller and Friedman, 2009) are a large class of Bayesian approaches including **Bayesian networks, Markov networks, random fields,** and more. Of these approaches, **Markov models** are particularly suitable for modeling temporal dynamics. In a Markov model, the data are assumed to come from a system with discrete states, and each state is assumed to have a different distribution for generating observations (Rabiner, 1989). The first order Markov assumption is that each state contains all the necessary historical information for making inference; if we know the current state, past observations do not add anything to our prediction capability. In **hidden Markov models** (HMM), the state is not known, but estimated along with observation distributions per state, as well as state transition probabilities. HMM are used in UC for several problems, such as identifying different types of road segments based on traffic information (Zheng et al., 2014).

3.5. Measuring the performance of ML approaches

The "training" of ML approaches, as well as their evaluation, is guided by several performance measures. Choosing the appropriate measure is important, as these measures highlight different aspects. For example, **classification accuracy** denotes the percentage of correct classification for an ML model, and is used for supervised classification tasks. For regression tasks, on the other hand, **mean squared error** can be used to denote the average distance from the true (expected) regression value. These can be reported separately on training, validation, and test sets. If cross-validation is used, it is typical to report the mean and standard deviation of a measure to illustrate how much it fluctuates across folds. For C classes, a CxC **confusion matrix** shows the labels assigned by the classifier vs. the true label, and thus shows which classes get confused with which other classes. This is useful to get some more insight. If multiple ML approaches are going to be compared on a specific problem, looking at the differences will not say much, and statistical testing should be used on these measures to select the best classifier (Alpaydın, 1999).

When the data are distributed to classes in an uneven way (also called **data imbalance**), then the dominant class will play a greater role in error measures. For example, in a security-cam based anomaly detection application, most footage will be labeled 'normal', and only very few moments are designated as 'anomaly'. If anomalous footage is 0.1% of all the footage, an ML system that always predicts 'normal' will have an accuracy of 99.9%. In order to deal with data imbalance, error is investigated in greater detail, and we distinguish between **false positives** (e.g. normal segment classified as anomaly), **false negatives** (e.g. anomaly classified as normal), **true positives** (e.g.

anomaly labeled as anomaly) and **true negatives** (e.g. normal labeled as normal). Then, **precision** is calculated as true positives over all positives, and for the example of anomaly detection, it is high if most of our anomaly labels are indeed anomalies. **Recall** is calculated by true positives over all samples which are actually positives (i.e. true positives plus false negatives), and shows the proportion of anomalies we were able to capture. The **F_1 score** is computed as $2\frac{Precision*Recall}{Precision+Recall}$, and balances these measures. For imbalanced classification problems, F_1 score is typically reported.

4. Further considerations in the design of ML approaches

The complexity of ML algorithms depends on the number of free parameters, as well as on the complexity of operations performed by the algorithm. Complex models need more training samples to train, otherwise **overfitting** will reduce generalization power of these models. In selecting an appropriate ML approach, we pay attention to the amount of data at hand, its feature dimensionality (which can be reduced with feature selection or extraction), as well as the nature of the data (e.g. spatial, temporal, spatiotemporal). For **unbalanced problems,** where we have a lot of samples from one class, but very few samples from another (e.g. accident and anomaly detection problems), statistical techniques are used to create richer samples for training. **Cross-validation** and **bootstrapping** are methods of dividing our dataset into training and validation sets to replicate the analysis in different samples. In K-fold cross validation, the data are divided into K equal sets, and each one of the parts are designated as the validation set once, while the rest is used in training. As a result we have a K different combination for validation and training sets. Bootstrapping, on the other hand, draws new samples with replacement for cross-validation.

To increase the predictive accuracy of ML models we can combine multiple learners; an approach called **ensemble learning**. Ensemble methods can be used for classification or regression, with parametric or non-parametric approach. The more diverse the **base-learners** are (i.e. making errors on different input samples), the higher the chance of getting a better result by ensemble learning. **Bagging** is an ensemble learning type that uses different training sets by randomly drawing different samples (Breiman, 1996). **Boosting**, on the other hand, sequentially trains the base-learners and in each step it emphasizes the training instances that were previously misclassified (Schapire, 1990). Boosting creates more complementarities between the base-learners to achieve higher accuracy but it also has a tendency of overfitting the data. The combination of multiple base-learners can be done parallely, like in **voting** and **stacking** (Wolpert, 1992), or serially, like in **cascading**.

The **no-free-lunch theorem** tells us that finding one single algorithm that can outperform other learners in any dataset is not an achievable goal (Wolpert and Macready, 1997). Though, by following certain steps we can still design ML processes so that we choose the learner that performs best on a validation set. While making inference from data with ML approaches, we often face factors that we can control, such as the algorithm itself, and method-specific hyperparameters, and factors that we cannot control, such as the noise in the data. It is common to test a range of values for the controllable factors, but cherry picking should be avoided lest the generalization power drops. However, other important aspects like cost of computation, and **interpretability** of the result should also be taken into consideration while comparing different machine learning models.

Most ML methods are not designed to show causal relationships in the data. The idea in traditional statistics is that the associations within the data cannot be interpreted from a **causal** lens, unless they are produced within an experimental setting. “Correlation is not causation” is a famous maxim in statistics, but as Judea Pearl (Pearl, 2020) puts nicely, statistics does not tell us what causation is. Without a model at hand to explain the process that generates the data, the data itself will not give us any causal relationship, but interpretability and explainability may be very important in

UC, as the predictions could have policy implications. Typical examples are criminal behavior prediction (Loyola-Gonzalez, 2019) or traffic congestion mitigation (Sun et al., 2019).

5. A brief word of conclusion

We have covered a very broad and rapidly progressing area in a very condensed form. The ML approaches we have discussed each have many variants and extensions. The most important concepts, however, are established and are strongly linked to long standing research in statistics, probability, and optimization theory. Clearly, ML-based prediction and analysis is a valuable tool for UC researchers, and the combination is receiving attention (Barlacchi, 2019), yet there are pitfalls of reductionism. Abstractions of ML can reduce people to numbers, and prediction rates and accuracies can create the illusion of understanding the dynamics of life in a city, but every person is unique and has a story, and algorithms should not be instruments to hide this fact.

Acknowledgments

This work is supported by the HumMingBird project, which has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No 870661. We thank Carlos Edmundo Arcila for his constructive comments.

References

Alpaydın, E. (1999). Combined 5×2 cv F test for comparing supervised classification learning algorithms. *Neural computation,* 11(8), 1885-1892.

Alpaydın, E. (2020). *Introduction to machine learning*. 4th ed., MIT Press.

Barlacchi, G. (2019). *Machine Learning Methods for Urban Computing* (Doctoral dissertation, University of Trento).

Bogomolov, A., Lepri, B., Staiano, J., Oliver, N., Pianesi, F., & Pentland, A. (2014). Once upon a crime: towards crime prediction from demographics and mobile data. In *Proceedings of the 16th international conference on multimodal interaction* (pp. 427-434).

Breiman, L. (1996). Bagging Predictors. *Machine Learning*, 24, 123–140.

Buntine, W. (1992). Learning classification trees. *Statistics and Computing*, 2(2), 63–73.

Chatfield, K., Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Return of the devil in the details: Delving deep into convolutional nets*BMVC 2014 - Proceedings of the British Machine Vision Conference 2014,* 1–11.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273–297.

Dempster, A., Laird, N., & Rubin, D. (1977). *Journal of the Royal Statistical Society. Series B.* Vol. 39, No. 1 (1977), pp. 1-38.

De Nadai, M., Vieriu, R. L., Zen, G., Dragicevic, S., Naik, N., Caraviello, M., ... & Lepri, B. (2016, October). Are safer looking neighborhoods more lively? A multimodal investigation into urban life. In *Proceedings of the 24th ACM international conference on Multimedia* (pp. 1127-1135).

Everett, B. (2013). *An introduction to latent variable models*. Springer Science & Business Media.

Ferreira, D., Nunes, B., Campos, C., & Obraczka, K. (2020). A Deep Learning Approach for Identifying User Communities Based on Geographical Preferences and Its Applications to Urban and Environmental Planning. *ACM Transactions on Spatial Algorithms and Systems*, 6(3).

Frias-Martinez, V., & Frias-Martinez, E. (2014). Spectral clustering for sensing urban land use using Twitter activity. *Engineering Applications of Artificial Intelligence* 35 237-245.

Gao, M., & Feng, Q. (2009). Modeling and forecasting of urban logistics demand based on support vector machine. *Proceedings - 2009 2nd International Workshop on Knowledge Discovery and Data Mining, WKKD 2009*, 793–796.

Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*, *521*(7553), 452-459.

Gianni Barlacchi (2019). *Machine Learning Methods for Urban Computing*. Doctoral dissertation, Universita Degli studi di Trento.

Giriraja, C., Haswanth, A., Srinivasa, C., Jaya Ram, T., Krishnaiah, P., & Manager, D. (2014). Satellite image classification using unsupervised learning and SIFT. *ACM International Conference Proceeding Series, 10-11-Octo*.

Gundogdu, D., Incel, O. D., Salah, A. A., & Lepri, B. (2016). Countrywide arrhythmia: emergency event detection using mobile phone data. *EPJ Data Science*, 5(1), 25.

Hastie, T., Friedman, J., & Tisbshirani, R. (2017). *The Elements of statistical learning: data mining, inference, and prediction*. New York: Springer.

Huang, C., Wang, D., & Tao, J. (2017). An unsupervised approach to inferring the localness of people using incomplete geotemporal online check-in data. *ACM Transactions on Intelligent Systems and Technology,* 8(6).

Jenkins, P., Farag, A., Wang, S., & Li, Z. (2019). Unsupervised representation learning of spatial data via multimodal embedding. *International Conference on Information and Knowledge Management, Proceedings*, 1993–2002.

Jiang, R., Song, X., Fan, Z., Xia, T., Chen, Q., Chen, Q., & Shibasaki, R. (2018). Deep ROI-Based Modeling for Urban Human Mobility Prediction. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(1), 1–29.

Jollife, I., & Cadima, J. (2016). Principal component analysis: A review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065).

Kobori, H., Dickinson, J. L., Washitani, I., Sakurai, R., Amano, T., Komatsu, N., ... & Miller-Rushing, A. J. (2016). Citizen science: a new approach to advance ecology, education, and conservation. *Ecological research*, *31*(1), 1-19.

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT Press.

Kuncheva, L. (2005). Diversity in multiple classifier systems. *Information Fusion*, 6(1), 3–4.

Kuncheva, L. (2014). *Combining pattern classifiers: methods and algorithms*. Wiley.

Li, M., Ch'ng, E., Chong, A., & See, S. (2016, July). The new eye of smart city: novel citizen sentiment analysis in Twitter. In *2016 International Conference on Audio, Language and Image Processing (ICALIP)* (pp. 557-562). IEEE.

Liu, J., Li, T., Xie, P., Du, S., Teng, F., & Yang, X. (2020). Urban big data fusion based on deep learning: An overview. *Information Fusion, 53(August),* 123–133.

Liu, X., Kang, C., Gong, L., & Liu, Y. (2016). Incorporating spatial interaction patterns in classifying and understanding urban land use. *International Journal of Geographical Information Science*, 30(2), 334–350.

Loyola-Gonzalalez, O. (2019). Understanding the Criminal Behavior in Mexico City through an Explainable Artificial Intelligence Model, *Lecture Notes in Computer Science*, 1, (pp. 136–149).

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2019). A survey on bias and fairness in machine learning. *arXiv preprint arXiv:1908.09635*.

McCulloch, W., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4), 115–133.

Mitchell, T. M. (1980). *The need for biases in learning generalizations* (pp. 184-191). New Jersey: Department of Computer Science, Laboratory for Computer Science Research, Rutgers Univ..

Pearl, J. (2020). *The Book of why: the new science of cause and effect*. BASIC Books.

Peng, H., Long, F., & Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, *27*(8), 1226-1238.

Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, 1(1), 81–106.

Quinlan, J. (1990). Learning Logical Definitions from Relations. *Machine Learning*, 5(3), 239–266.

Richard A . Redner, & Homer F . Walker (1984). *SIAM Review*. Vol. 26, No. 2 (Apr., 1984), pp. 195-239.

Rosenblatt F (1961) *Principles of neurodynamics: Perceptions and the theory of brain mechanism*. Spartan Books. Washington, DC.

Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088), 533–536.

Salah, A. A., Pentland, A., Lepri, B., & Letouzé, E. (2019). *Guide to Mobile Data Analytics in Refugee Scenarios*. Springer, Berlin.

Saldana-Perez, M., Torres-Ruiz, M., & Moreno-Ibarra, M. (2019). Geospatial Modeling of Road Traffic Using a Semi-Supervised Regression Algorithm. *IEEE Access*, 7, 177376–177386.

Santani, D., Ruiz-Correa, S., & Gatica-Perez, D. (2015, December). Looking at cities in Mexico with crowds. In *Proceedings of the 2015 Annual Symposium on Computing for Development* (pp. 127-135).

Schapire, R. (1990). The Strength of Weak Learnability. *Machine Learning*, 5(2), 197–227.

Sun, J., Guo, J., Wu, X., Zhu, Q., Wu, D., Xian, K. & Zhou, X. (2019, May). Analyzing the impact of traffic congestion mitigation: From an explainable neural network learning framework to marginal effect analyses. Sensors (Switzerland), 19(10).

Toole, J., Ulm, M., González, M., & Bauer, D. (2012). Inferring land use from mobile phone activity. *Proceedings of the ACM SIGKDDInternational Conference on Knowledge Discovery and Data Mining*, 1–8.

Wolpert, D. (1992). Stacked Generalization (Stacking). *Neural Networks*, 5, 241–259.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, *1*(1), 67-82.

Yang, J., & Eickhoff, C. (2018). Unsupervised learning of parsimonious general-purpose embeddings for user and location modeling. *ACM Transactions on Information Systems*, 36(3).

Yuan, J., Zheng, Y., & Xie, X. (2012). Discovering regions of different functions in a city using human mobility and POIs. *Proceedings of 18th SIGKDD Conference on Knowledge Discovery and Data Mining,* 186–194.

Zheng, K., Zheng, Y., Yuan, N. J., & Shang, S. (2013, April). On discovery of gathering patterns from trajectories. In *2013 IEEE 29th international conference on data engineering (ICDE)* (pp. 242-253). IEEE.

Zheng, Y., Capra, L., Wolfson, O., & Yang, H. (2014). UC: Concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology*, 5(3).