Introduction to Machine Learning (for AI)

Introduction to Machine Learning

Dr. Matias Valdenegro

November 14, 2024

Today's Lecture

- In this lecture you will learn the basic Machine Learning concepts.
- We will see the three paradigms of ML: Supervised, Unsupervised, and Reinforcement Learning.
- We will discover what are features and a bit on how to engineer them...
- And you will learn basic data preprocessing and normalization applied to features, together with the biggest challenges in ML.

Outline

Introduction and Basic ML concepts

- Machine learning is a field that studies algorithms that can learn from data.
- In this part of the lecture we will cover the basic concepts in detail.
- In this course we will cover classical ML algorithms, which are the base for more complex algorithms like neural networks. The basic concepts are the same, so covering them in detail is very important.

Introduction and Basic ML concepts

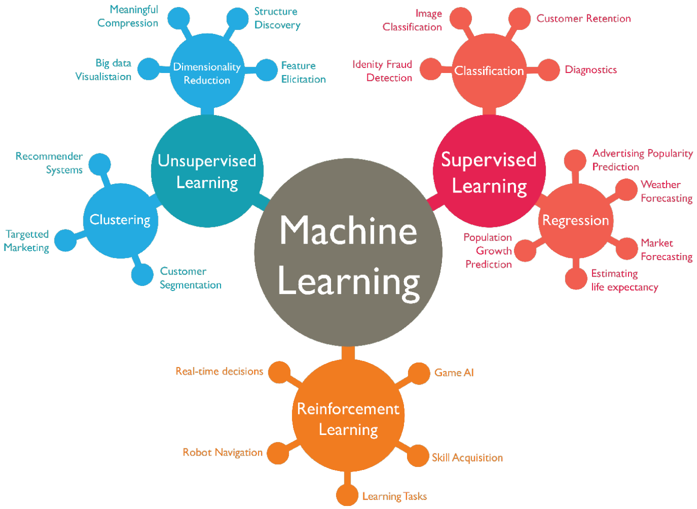


Image from `https://7wdata.be/big-data/10-companies-using-machine-learning-in-cool-ways/`

What is required to reach True AGI?

Adaptivity or Adaptability

Be able to learn from previous errors and improve overall performance by learning from experience.

Learning

Learning is the process of acquiring new understanding, knowledge, behaviors, skills, values, attitudes, and preferences.

Generalization

Lessons and learned knowledge should be general enough that they also work in (new) previously unseen environments.

The Qualification Problem

Modeling the real world is very difficult. The Qualification problem is that it is impossible to enumerate or list all pre-conditions that are required for an action in the real world to succeed and have the intended effect. For example:

- To attend an online lecture, you need to have an internet connection.
- You also need a power source for your computer.
- And the computer should not be broken.
- And the network interface should also work correctly.
- And so on...

Actions can also have unintended consequences, and enumerating these is also impossible.

Data Driven Paradigms

In our time, the AI field is a combination of numerous techniques. The (currently) dominant set of techniques is machine learning, meaning that algorithms learn from data and/or experience.

Large quantities of data are used to specify a task that the agent should learn. For some tasks, this data might require labels provided by a human, while other tasks might learn without labels.

This alleviates the Qualification problem, as data is better (but not perfect) in capturing the realities of our world.

Learning Paradigms

Supervised Learning

In supervised learning, the agent learns through supervision, that is, a teacher tells the agent what is the correct answer given some input or stimuli, and the agent should learn the mapping between input and desired output.

Supervised Learning is generally limited by the quantities of data that are labeled.

Producing labels for datasets is expensive (in time and money), and in many cases there might not be a correct answer to provide a label.

Learning Paradigms

Unsupervised Learning

In Unsupervised Learning, with contrast of Supervised Learning, the agent learns from data directly, without any kind of supervision. There is no correct answer. The agent should learn structure, properties, and relationships in the data itself.

Unsupervised Learning is not limited by labels, thus it can exploit large amounts of data that are not labeled, the only limitation is how to capture additional data of interest.

The problem in Unsupervised Learning is generally algorithmic, it is difficult to extract patterns and properties of data without supervision.

The ideal Unsupervised Learning method would make no assumptions on data.

Learning Paradigms

Unsupervised Learning

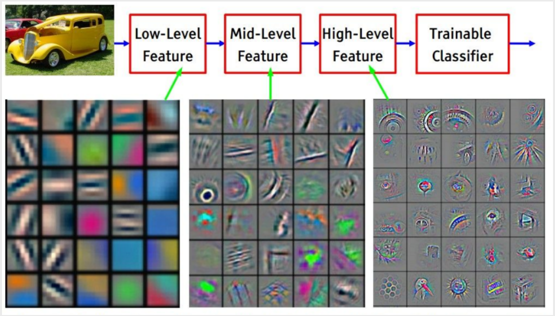


Figure: Example of convolutional features that can be learned using unsupervised learning on images. The algorithm learns that images consists of edges, then groups edges into parts, and groups them into more complex objects.

Learning Paradigms

Reinforcement Learning

In Reinforcement Learning, the agent interacts with an environment, sensing and performing actions that change the environment, and learns with a reward signal. This signal tells the agent how it is performing, but does not explicitly say which is the correct answer or the best action.

The agent learns a policy that maximizes a specified criterion, for example, to maximize the sum of rewards over a whole interaction episode.

This paradigm is commonly used for sequential decision making tasks, for example, controlling a robot to pick and object and place it somewhere else, or play video games.

Learning Paradigms

Reinforcement Learning

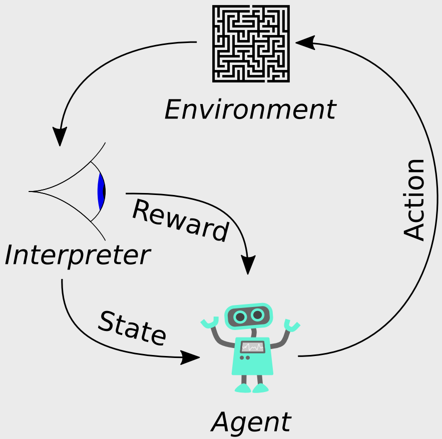


Figure: Diagram representing how the agent interacts with the environment, by performing an action, and then receiving a reward and a new state observation. The process is repeated sequentially until a final state is reached, or the episode ends. The agent learns through multiple episodes.

Figure from https://en.wikipedia.org/wiki/Reinforcement_learning

Learning Paradigms

Transfer Learning

It is a combination of previous paradigms, where learned knowledge is used to learn or improve on new tasks. Generally this means part of the learned knowledge transfers from previous tasks into new ones.

This is possible because tasks usually have some common knowledge that can be exploited. For example, in an image understanding problem, if one learns a image classifier for dogs/cats, the features that are learned in this case are also useful for classifying birds, as images naturally have features like edges, parts, etc.

Learning Paradigms

Combinations of Previous Paradigms

Self-Supervised
The agent/algorithm supervises itself by means of solving a pretext task. This task can be automatically computed and solved, but generally it needs to be designed by humans.

Semi-Supervised Learning
The agent/algorithm receives multiple kinds of supervision at the same time, for example it could be doing unsupervised learning on a set of images, while using that knowledge and performing supervised learning on a different set of images.

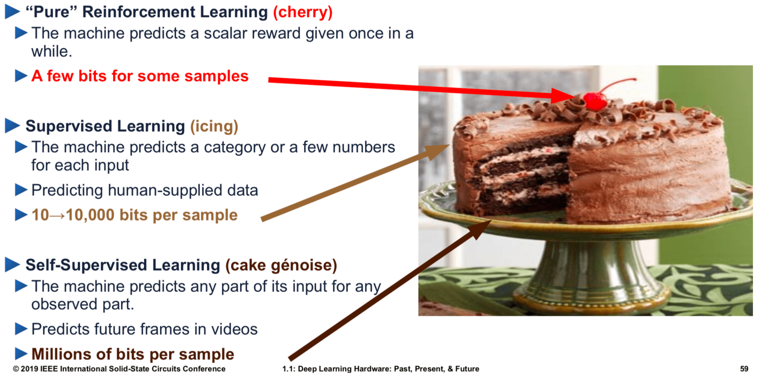
Learning Paradigms

Combinations of Previous Paradigms

Weak Supervision

The agent receives supervision but it is *weak*, meaning that the kind of supervision received is not enough to completely solve the task.
For example for object detection, bounding box labels are needed (box coordinates and class information), and in weakly supervised object detection, only class labels are supervised.

The Cake/Lecake - Relative Importance of Learning Paradigms



- **"Pure" Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Self-Supervised Learning (cake génoise)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

© 2019 IEEE International Solid-State Circuits Conference 1.1: Deep Learning Hardware: Past, Present, & Future 59

Figure: The Lecake, A common representation on the value or relative importance of each learning paradigm, made by Yann LeCun.

What is **not** Machine Learning?

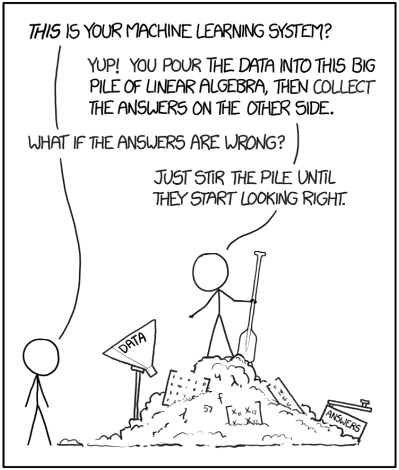

THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

Image from `https://xkcd.com/1838/`

What is **not** Machine Learning?

- Machine Learning is **not magic**, data is not the new oil/electricity, and proper methods still need to be developed.
- It is a rigorous field that is more than just applying methods/libraries without thinking.
- If methods or evaluation are not performed correctly, unrealistic performance can be obtained. Then you only fool yourself.
- Many startups expect Machine Learning to magically solve their problems. But it does not work that way.

Core Idea of Machine Learning

Machine Learning, as a field, has a bunch of methods, techniques, methodologies, but in the end the core idea or most important concept is:

Generalization

This means you train a model on a dataset, and it should work (make correct predictions) on very different input data, be useful outside of its training space.

We will define generalization more formally in coming lectures.

Outline

Basic ML concepts

Model

A model is the actual equations that define how the input is transformed into output predictions. For example:

$$f(x) = \sum_i w_i x_i + b$$

We denote models as $f(x)$ where x is the input, and models with probability as $p(y|x)$. Note that ML models are usually predictive ones.

Parameters or Weights

These are the variables that change during the training process, implicitly encoding the knowledge that is learned. We usually denote these with letters w and b (weights and biases).

Basic ML concepts

Loss Function

A function that makes a comparison between the model's prediction and the ground truth labels. It provides direct supervision to the model so it can learn successfully. Loss functions are selected to learn a particular task, in coming lectures we will cover the full details.

Training Data

The data that you use to train the model, represented mathematically as the x and y variables, where x represents inputs, and y are the labels associated with that input.

Basic ML Concepts

Classification

Classification is the task where the output variable and/or labels are *discrete*, that is, the opposite of continuous. A model that performs classification is called a classifier.

The set of possible outputs for the classifier is called the class set, and the individual elements are called classes.

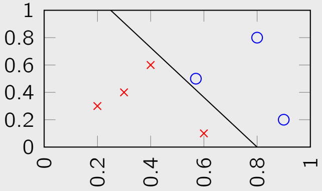


Figure: Binary Classification (Two classes)

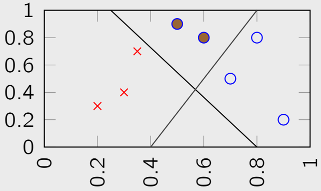


Figure: Multi-Class Classification (>2 classes)

Basic ML Concepts

Regression

Regression is the task where the output variable and/or labels are *continuous*. A model that performs regression is called a regressor.

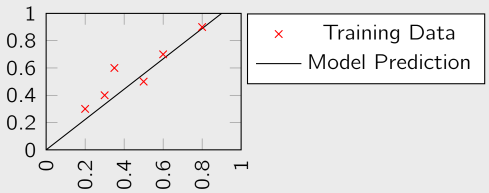


Figure: Regression in 2D

Basic ML Concepts

Multi-Task Learning

- It is also possible to perform/learn more than one task at a time.
- For example, performing object detection requires object recognition (classification) and bounding box estimation (regression).
- The idea is that tasks have some input or features in common, so performance is higher when learning these tasks together, than each individual task in isolation.

Learning as Optimization

All machine learning models learn from data, using a general formulation that consists of the following optimization problem:

$$\theta^* = \min_\theta \sum_i^n L(f(x_i), y_i, \theta) \tag{1}$$

The solution to the optimization problem is θ^*, the weights/parameters that produce the best predictions, according to what the loss function scores between the prediction $f(x_i)$ and the labels y_i.

The loss function is usually designed to be used to learn a specific task. A common source of errors is choosing the wrong loss function.

Why does this work?

- This might look as simple function approximation (it is), but when the model is a complex non-linear function, and the training data is a large dataset, they can contain and represent complex relationships that are not easy for humans.
- The performance of a model increases with the number of data points in the training set. We will go deeper in this direction in future lectures.
- In any case this requires a considerable amount of engineering work, including capturing and labeling data, selecting preprocessing pipelines and models, and training.

Trainable and Non-Trainable Parameters

In the previous formulation, we usually train the parameters using an optimization algorithm, but there could be multiple types of parameters.

Trainable Parameters

Parameters that are directly trained using an optimization algorithm.

Non-Trainable Parameters

Parameters that are trained but not using an optimization algorithm, for example, models that indirectly learn parameters from data that passed through them.

In transfer learning, parameters can also be frozen, where they are kept constant, and new models are trained on top of learned features. This is done so knowledge is preserved.

Train and Test Datasets

Datasets are usually used for multiple purposes, like training and evaluation. This requires that once you have a set of data points, it needs to be split into at least two datasets:

Training Set

The set that will be used for training using the optimization process, it usually is split from around 70% to 80% of the data available.

Test Set

The set that will be used for evaluation of the trained model, it should be a independent dataset containing samples that have not been seen during training. It can be the remaining 20% to 30% of the available data.

Linear Separability

This is important as the basic classification concept uses separating hyperplanes.

Definition

Given points $x_i \in \mathbb{R}^n$ in a high dimensional space (features), with binary labels $y_i \in [0, 1]$ they are linearly separable if there exists a weight vector $\mathbf{w} \in \mathbb{R}^n$ and real number k such as:

$$\begin{aligned} \mathbf{w} \cdot x_i > k \qquad & \forall x_i \in [x_i \,|\, y_i = 0] \\ \mathbf{w} \cdot x_i < k \qquad & \forall x_i \in [x_i \,|\, y_i = 1] \end{aligned}$$

In simple words, there is a hyperplane that separates both classes $y = 0$ and $y = 1$ in the feature space.

Note that the separating hyperplane $\mathbf{w}$ does not have to be unique. This concept can be extended for > 2 classes.

Linear Separability

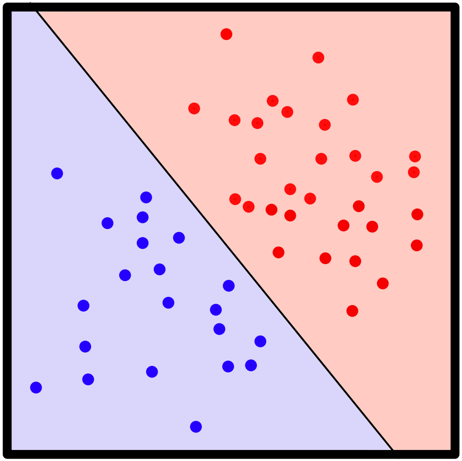


Figure from https://en.wikipedia.org/wiki/Linear_separability

Blue points are class 0, and red points are class 1.

Not Linearly Separable

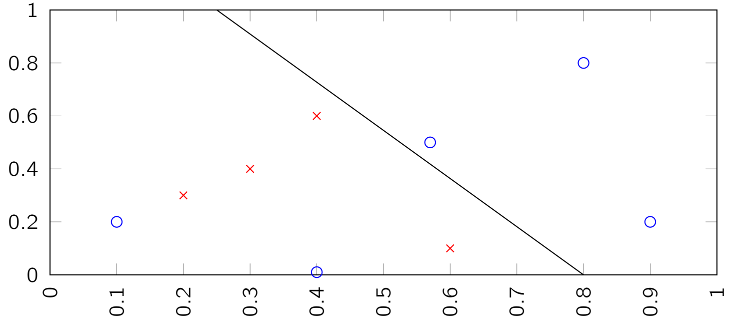


Figure: Example of Not linearly separable data for Binary Classification.

The role of Feature Engineering for Classification is to transform features to be linearly separable.

Outline

Basic ML Concepts

Features

This is a very important and highly misunderstood term. A feature is a measurement, measurable property, or characteristic of a phenomenon or physical object.

The inputs to a machine learning model are feature values, that are used to perform classification or regression. The *quality* of the features is what defines the performance of your model.

- Good features will lead to good performance of your model.
- Bad features will prevent your model from making correct predictions. Model might not train at all.
- Features should be de-correlated and independent for best results. If features are related to each other, then they are redundant and not useful for learning a model.

Basic ML Concepts

Features - Examples

Images
: Pixel values, edges, parts, color(s), distribution of pixel values (histograms), distribution of edges, gradient of image.

Text
: Frequency of words, language, presence or absence of specific words, # of grammatical errors, overall

Sound
: Frequency response (Fourier transform coefficients), response to filter banks, signal to noise ratio, length of sound sequences.

Note that features should be representable as numerical data.

Basic ML Concepts

Feature Vectors

To finalize the concept of a feature, we need to talk about the concept of a feature vector. This is a vector with elements corresponding to individual features. For example let's try to predict the rent for an apartment, our features could be:

A. Area in m^2 (continuous).

B. Location in the city (categorical).

C. Distance to nearest train station (continuous).

D. Build year (categorical).

Then a feature vector could be $\mathbf{x} = [A, B, C]$, or $\mathbf{x} = [A, C, D]$, or $\mathbf{x} = [A, D]$, etc.

Designing features and feature vectors is called feature engineering.

Types of Features

Continuous

Real numbers. For example, distances, floating point numbers, convolutional feature maps, gradients, etc.

Discrete

Integer numbers that can be ordered and are countable. For example, number of shoes in your apartment, most currencies (€, $), number of persons living in a house, population of a country, etc.

Categorical

Categories, where no order can be established. For example, colors, nationality, course code, etc.

Obtaining Features

There is a whole field of Feature Engineering, to obtain high quality features for tasks. Specially in NLP and Computer Vision.

The quality of your features related to a specific task, defines the performance of your machine learning model. Better features means better performance, and viceversa.

Features can also be (non-linear) transformations of raw data (pixels, sound, text, etc). This is what Deep Learning does.

Feature Transformations

A simple way to "improve" your features is to make aggregation or non-linear transformations of features. For example:

Aggregation
: Mean, standard deviation, variance, skew, kurtosis, etc.

Polynomials
: For a scalar feature x, use $x, x^2, x^3, x^3, ..., x^p$. The polynomial degree p is a hyper-parameter.

Frequency
: Fourier transform of your data is commonly used as features, with a frequency cutoff.

Not Useful Features

- Features that are linearly correlated with each other.
- Features that do not change over the whole dataset (constant).
- Features with many missing or unclear values.
- Features that do not predict the target value of interest.
- Random features.

Outline

Pre-Processing

In general data needs to be pre-processed and normalized before being used with any model. This is because when learning a model, it will only work with data that is similar to the training data distribution, so normalizing allows data to be more similar to this distribution, and not to vary too much.

- Pre-processing can also discard invalid data, select which features will be used, and put data in a common format.
- Train and test splits are also part of pre-processing.
- Depending on the task (like regression), outputs can also be post-processed to produce ideal predictions.

Normalization

Normalization is the process where input features are standarized to a common range.

- This is required as features might be at different scales (numerical range), and the model might assign importance to each feature according to the highest value.
- In general training a model with normalized data is much easier, so it is a recommended first step.
- Target values can also be normalized, for example, to make regression easier (by limiting the model output range).
- Intuition is that each (normalized) feature should have similar range.

Normalization

- We cover a set of normalization techniques. A *very important* point is that normalization generally captures some statistics from the training set, so any normalization should be applied in the training set, and then in the test set (using train statistics) and in any new data when making predictions.
- Normalization allows all features to have similar ranges, which makes training smoother. Models trained on unnormalized data usually perform worse.
- Sometimes it is difficult to normalize when the training and test sets vary considerably.

Normalization

Min-Max

For each feature x, compute the following:

$$x_{norm} = \frac{x - \min x}{\max x - \min x} \tag{2}$$

This transforms all values to range inside $[0, 1]$. The minimum feature value is mapped to zero, and the maximum feature is mapped to one. Intermediate values are mapped to $(0, 1)$.

There is also a transformation to the $[a, b]$ range:

$$x_{norm} = (b - a)\frac{x - \min x}{\max x - \min x} + a \tag{3}$$

Normalization

Standard or Mean-Std

For each feature x, compute the following:

$$z = \frac{x - \text{mean}(x)}{\text{std}(x)} = \frac{x - \mu_x}{\sigma_x} \tag{4}$$

This transforms all values to an approximate range, usually $[-3, 3]$, depending on the distribution of x. You might recognize that this is the same as the z-score used in a normal/gaussian distribution. The mean of z is 0.0, and the standard deviation and variance is 1.0.

Normalization

Unit Vector

For a feature vector $\mathbf{x}$, compute the following:

$$v = \frac{\mathbf{x}}{||\mathbf{x}||} = \{\frac{x_i}{||\mathbf{x}||}\}_i \tag{5}$$

This makes the data lie on a n dimensional hyper-sphere. Where $||\mathbf{x}||$ is a vector norm, for example the L2 norm $||\mathbf{x}|| = \sqrt{\sum_i x_i^2}$

Normalization - Whitening

Whitening is a more complex transformation of multi-dimensional data X, with the purpose of de-correlating and setting the covariance matrix Σ of the data to the identity matrix.

For zero-centered (mean substracted) data X, the basic transformation is:

$$Z = WX \tag{6}$$

The matrix W is called a whitening matrix, and should follow that $W^T W = \Sigma^{-1}$.

Normalization - Whitening

Depending on the choice of W, there are different types of whitening:

ZCA Whitening

Select $W = \Sigma^{-\frac{1}{2}}$

PCA Whitening

Compute W using Principal Component Analysis (PCA).

Note that ZCA and PCA whitening produce different results but in both cases the data is normalized to identity covariance.

Normalization - PCA Whitening

Here the matrix W is computed using the eigenvalue decomposition on Σ, which is a matrix decomposition in the form:

$$\Sigma = U \Lambda U^T \tag{7}$$

Where U contains the eigenvectors of Σ as columns, and Λ is a diagonal matrix containing the eigenvalues of Σ. This can be computed using functions like `np.linalg.eig`. Then W is computed as:

$$W = \Lambda^{-\frac{1}{2}} U^T \tag{8}$$

This sets the covariance of X to be a identity matrix.

Normalization - Images

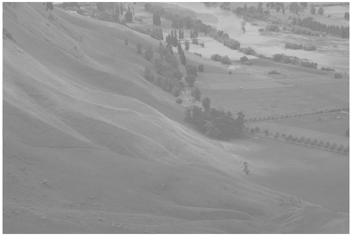
For specialized data types, there are specific normalization methods. For images we have:

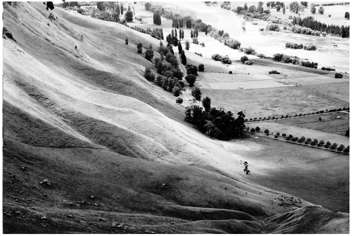
Histogram Equalization This adjust contrast in the image so the distribution (histogram) of gray levels is uniform. This makes sure that all gray levels are equally represented in the normalized image. It is useful for when the image has dominant shadows or highlights.

Normalization - Images

Noise Removal	Noise can be removed using a median or gaussian filter, with a careful design to not remove important image features.
Resizing	Images in a dataset can vary in size, and they are usually normalized to a standard size. Common values are 224×224 or 256×256. Normalized size must be carefully chosen to not lose image features.

Normalization - Histogram Equalization





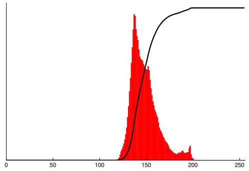


Figure: Original image and histogram (red). The black curve is the cumulative distribution.

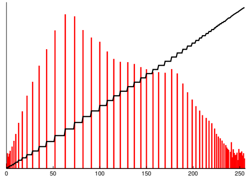


Figure: Equalized image and histogram (red). The black curve is the cumulative distribution.

Normalization - Audio

Resampling The sampling rate of the audio signal can also be normalized, common values at 44.1 kHz or 22.05 kHz.

Audio Length The length (in time) of audio signals can also be normalized, in a similar way to image size normalization.

Normalization - Audio

Amplitude Normalization
: The overall volume of audio can also be normalized, to make sure all samples in the dataset have similar volume. This is done in two ways:

1. **Peak Normalization**. Normalizing to the maximum value across time samples. This is similar to min-max scaling.
2. **Loudness Normalization**. Normalizing to a aggregate loudness of the whole audio signal across time. For example it can be the root mean square (RMS), or a value related to human perception of loudness.

Normalization - Text

Text is the most difficult to normalize, since many factors can vary:

- Stop words are usually removed.
- Numbers, dates, acronyms and abbreviations can be standarized to common terms (like transforming 10 to Ten, dates to ISO formats, and acronyms removed or expanded.
- Non alphanumeric characters are usually removed.
- Specific words can be standardized, specially if they have multiple meanings, depending on the application.
- Spelling across variations of the same language can also be normalized, for example British vs Australian vs American English.

Normalization - Categorical Variables

Categorical variables are usually not numeric, for example, when a variable can take a value from a pre-defined set of values. A common mistake is to just assign numerical values to each of the categorical values, but this biases the model, as it thinks higher value means more importance.

So you **cannot** use categorical variables by converting them to integers. This is a **common** mistake.

Normalization - Categorical Variables

An unbiased way to encode categorical variables is to use one-hot encoding.

If the variable has N possible values and we encode category i, then a vector of N elements is used to represent it, where all elements are 0.0, except for the i-th value which is set to 1.0.

An example:

- Cat $= [1.0, 0.0, 0.0]$
- Dog $= [0.0, 1.0, 0.0]$
- Bunny $= [0.0, 0.0, 1.0]$

Normalization - Regression

One important detail, is that output values (labels) can also be normalized, particularly in the case of regression.

- If your labels have a limited range, normalizing can teach the model what values are allowed. Models can also be configured to output in a normalized range.
- One should also make sure to remove or clip output values that are outside of the expected ranges.
- Correct normalization of the output and labels can the process of learning a model much easier.

Normalization - Which to use?

We have covered a lot of normalization methods. Choosing them is difficult, usually we experiment during ML model development and choose what performs best.

- In many cases it is trial and error.
- Experience tells us what to try first.
- Exploratory analysis of your data can reveal properties that should be normalized. Always look at your data before trying to learn models from it.

Normalization - Importance

What can happen if you do not normalize your data?

- Model underperforms.
- Training fails (loss does not decrease and metrics do not improve).
- Model makes prediction that do not make sense, specially if.
- Much higher chance of getting stuck in local minima.

The loss surface/landscape and the random weight initialization are both affected by normalization of the input/output.

Normalization - (Negative) Example

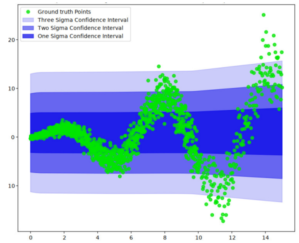


Figure: No Normalization

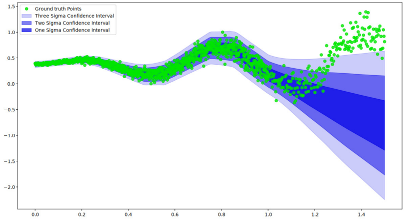


Figure: X and Y are normalized.

Notice the difference in fitting a non-linear model without (left) and with (right) normalization applied. It is literally night and day. The model shown here has uncertainty but the principle is the same.

Outline

Challenges in Machine Learning

Quality of Data

The biggest bottleneck is obtaining high quality data (including features and labels) for the task of interest. Without data, a model cannot be learned.

Low quality data (incorrect labels, unclear or useless features, missing values, etc) hurts the performance of your model.

Well known saying, Garbage in, Garbage out.

Challenges in Machine Learning

Model Misspecification

Model misspecification is using the *wrong* model for a given dataset.

In simple words, you have some features/data, a task and a target to predict. We **do not** know what would be the best model for this task.

In general you do not know the mathematical model that generated the data. Specially for highly dimensional data.

So we usually do trial and error and research to find out this.

Challenges in Machine Learning

Generalization

Overall I think the biggest issue currently in AI is the lack of generalization, that means, solutions are given to specific problems, but general AI solutions (true AGI) does not exist yet.

- Alpha Go and variations (Alpha Go Zero and Alpha Zero) only solved playing the game of Go, but these advances have not benefited other applications.
- Deep Blue can play chess, but it has not served to solve some of humanities biggest issues (like peace, hunger, wealth distribution, etc).
- All models overfit to some degree, but some are useful.

Challenges in Machine Learning

Explainability

Most AI methods overall are *Black Boxes*, meaning it is difficult to interpret their decisions, or to get an explanation on how a decision is made.

- Explanations and interpretability are required for AI to gain human trust, and it is even legally required in some countries (like in the EU with the GDPR).
- Explaining black box method is very difficult, as computational concepts do not always map to human understandable concepts. The use of non-linear methods also makes it difficult.
- Building methods that are easily interpretable (AKA white box methods) is also very difficult, and in many cases they work but are less accurate than black box models.

Challenges in Machine Learning

Bias, Safety, and Trustworthiness

As AI is used for real world applications, specially ones involving humans, it is very important to ensure that it will not perform unexpected actions or behave abnormally. Right now there are AI systems that are actively hurting people, for example:

- In the US, poorly understood AI systems are being used to decide if a person will commit a crime, or how long their sentence should be.
- AI systems being used for monitoring of populations, violating their human rights.
- AI systems used by banks to review and recommend on decisions for loans, and by companies to decide who to interview for a job.
- Discrimination by DUO and the Toeslagenaffaire (Benefits scandal).

Questions to Think About

1. What is a feature?
2. Describe linear separability in your own words.
3. What is the need for normalization and pre-processing of features?
4. Can regression labels/targets be normalized?
5. What factors can hurt the performance of a Machine Learning model?
6. Your model does not learn, what is the most likely cause?

Take Home Messages

- There are three paradigms in Machine Learning, Supervised, Unsupervised, and Reinforcement.
- Generalization is the biggest issue/core idea of Machine Learning.
- Two main tasks: Classification and Regression.
- Learning is done through an optimization problem.
- Normalization and pre-processing of features and labels is very important.
- Machine Learning is not magic, there are many challenges.

Questions?