1. Introduction

ML process of training a piece of software, called model, to make useful predictions or to generate content from data.

Types:
- Supervised learning(two most common use cases - regression and classification)
- Unsupervised learning(clusterization common)
- Reinforcement learning(penalties and rewards->generated policy)
- Generative AI(generate something from input)

Online and batch learning.

Y - the variable that we predict.

Feature(x) the variable in the data vector. Types:
1. Numerical
2. Categorical
   - Ordinal
   - Nominal

Hyperparam meta parameter for model. Model do not learn it.

1.1. Data mining

Data mining applying ML techniques to dig into large amounts of data can help discover factors, that are not immediately apparent.

1.2. Supervised

Solves regression and classification tasks.

$$X \longrightarrow F \longrightarrow y$$

Regression model predicts continuous values.
Classification model predicts categorical values.

Tasks:
1. Clustering
2. Anomaly detection
3. Dimensionality reduction
4. Association rule learning

Problems:
1. Underfitting/overfitting
2. Too few data/non representative data

1.3. Unsupervised

$$X \longrightarrow F \longrightarrow X'$$

1.4. Reinforement learning

The learning system, called agent can observe the environment, select and perform actions, and get rewards or penalties. It must learn then by itself what is the best strategy, called a policy to get the most reward over the time.

2. Four types of problems where it shines

1. Problems for which solution requires a lot of hand-tuning
2. Complex problems, for which there is no good solution exists
3. Fluctuating environments - online models can adapt
4. Getting insights about large amount of data.

2.1. Other differentiations

Batch vs live.

Instance-Based vs Model-Based learning.

Instance-based postponed computations, no training process - can produce output only in training dataset range.

Algorithms:
- K-neighbors regression(need to define neighborhood function)

Model-Based learning makes model to generalize data and predict new value by formula. Uses utility function to tweak the formula.

Algorithms:
- Linear regression

3. Main challenges

3.1. Bad data

1. Insufficient quantity of training data.

To detect use model performance metrics - precision/recall and cross validation.

2. Non representative training data.

Representative dataset - accurely reflect structure and the diversity of the real-world scenario.

To detect: use statistical analysis(check features distribution, outliners, tendencies etc.). Compare distribution of features with larger dataset.
- For classification check classes distribution.

3. Poor quality data(NULLS especially). Need to define how to treat outliners and NULLS.

3.2. Feature engineering

1. Feature selection
2. Feature extraction(e.g. compression)
3. Creating new features

3.3. On text procesing. (the unreasonable effectiveness of data)

The large amount of data with simple algorithms beats the small-middle amount of data with sophisticated algorithms.

The translation is a hard problem, but amount of data produces very precise prediction.

Actual problem of text processing is not to use statistics or ontology - but which language to use, how to feed info to it and how to run inference.

The ontology and other structuring is not possible at this scale because the profit is not obvious and not profitable. So at large scale it's more efficient to use unsupervised algorithms on unlabeled data.

3.4. Cross validation

Goal is to test model ability to predict new data by.(struggling with overfitting and selection bias)

Validation performed with different portions of data to test and train model.

3.5. Selection bias

Selecting individuals in such a way that proper randomization(representation) is not achieved.

4. Optimisation and loss function

4.1. Gradient descent

Gradient(∇f) defines direction and rate of fastest increase of scalar-valued differentiable function f.

Example for gradient in cartesian coordinate system f:

$$\nabla f = \frac{\partial f}{\partial x} i + \frac{\partial f}{\partial y} i + \frac{\partial f}{\partial z} k$$

Gradient descent iterative optimization algorithm of the first order to find the local minimum of the function.

Stop criteria for the gradient descent can be a threshold for the gradient value.

4.2. Optimization

Optimisation target minimize loss function.

Simple example of the loss function is a MSE.
Mean squared error(MSE) measures the average of squeared errors.

$$\text{MSE} = \frac{1}{N} \sum_{(x,y)\in D} (y - \text{prediction}(x))^2$$

Iteration step for model parameter:

$$\Theta^{i+1} = \Theta^i - h\frac{\partial f}{\partial \Theta^i}$$

Where h is a *learning step*.

How much i's would be in a N dataset?

Depends, upon model converges. Possible data slices for 1 ephoch:
- simple - 1 full dataset
- stochastic - 1 record
- mini-batch - batch of random examples(e.g. 10-1000). This approach can support struggling with local minimums.

TODO

[Comparison of batch sizes link]

epoch one pass of all the training examples
batch size the number of training examples in one pass. The higher the batch size, the more memory space you'll need.
iterations number of batches in epoch. each iteration adjusts model's parameters.

$$\frac{\partial f}{\partial \Theta_i} = \frac{1}{2N} \sum_{i=1}^{n} \left(\left(\sum_{j=1}^{m} (\Theta_j x_j) - y_i \right)^2 \right)'$$

note: N is the iteration dataset(or batch) size, x_j is a point in vector, Θj is the parameter value that is const if not differentiated, y_i is a constant for each i.

Let's simplify function for two parameters and 3 data slices:

$$\frac{1}{2N} \sum_{i=1}^{3} \left(\left(\sum_{j=1}^{2} (\Theta_j x_j) - y_i \right)^2 \right)$$

Simplify each i argument:

$$\left(\sum_{j=1}^{2} (\Theta_j x_j) - y_i \right)^2 = (\Theta_0 x_0 + \Theta_1 x_1 - y_i)^2$$

$\Theta_1 x_1$ and y_i is a constants if we differentiate by Θ_0, so we have: $\left((\Theta_0 x_0 + C_i)^2\right)'$, also: $(\Theta_0 x_0 + C_i)$ is an inside f v.

With *formula of compound derivative* $(u(v))' = u'(v) * v'$

$$\left((\Theta_0 x_0 + C_i)^2\right)' = 2(\Theta_0 x_0 + C_i)(x_0)$$

TODO

Final differential formula:

$$\frac{\partial f}{\partial \Theta_i} = x_i \frac{1}{N} \sum_{i=1}^{n} \left(\sum_{j=1}^{m} \Theta_j x_j - y_i \right)$$

Process is simple, count *gradient* for each *parameter* and change parameters by gradient descent.

$$\Theta_i \longrightarrow \text{gradient} \longrightarrow \Theta_i^T$$

4.3. Linear regression

When we have not linear plot, to solve this linear regression problem we can add additional polynomial(x^2) or functional($\sin(x)$, $\sqrt{x}$) features.

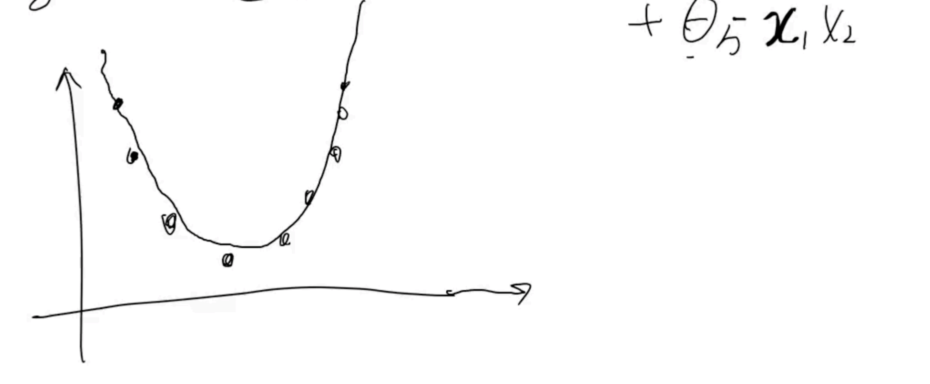

Figure 1: Synthetic features for regression with linear Θ params

How to choose function to create additional features? Intuitively as a hyperparams. There are automatic methods to make models - feature selection approach.

4.3.1. Normal Linear Regression Model

TODO https://www.statlect.com/fundamentals-of-statistics/normal-linear-regression-model

5. Data

Dataset should be divided minimum for train(60), validation(20) and final test(20). This divided datasets must not have semantic intersections(same people, same cars, same buildings etc.).

Cross validation - method, which on small dataset find conceptual ML model that possibly solves task. TODO

5.1. Underfitting and overfitting

Underfitting model performs poorly

Causes: Model is too weak How to beat: Make model more complex

Overfitting model performs well on training data, but not in evaluation

Causes: To complex model, too few data How to beat: Simplify model, add data

Problem is: Bias vs variance tradeoff.

Regulaization technique of discouraging learning a more complex or flexible model, so as to avoid the risk of overfitting.

bias

variance

Regularization formula L2(makes features balance):

$$L = \frac{1}{2N} \sum_{j=1}^{n} (y(x_i) - y_i)^2 + \Lambda \sum_{i=1}^{m} (\Theta_i^2) \longrightarrow \min_{\Theta}$$

Regularization formula L1(makes feature selection):

$$L = \frac{1}{2N} \sum_{j=1}^{n} (y(x_i) - y_i)^2 + \Lambda \sum_{i=1}^{m} (|\Theta|) \longrightarrow \min_{\Theta}$$

6. Regressions comparison

Criteria	Linear regression	Logistic
Regression plot dependency	Straight line	S-shaped curve
Output type	Continuous	Probability (0,1) of the value from the finite category
Target	Give a most precise number	Give a probability of belonging to category
Usecase	Predict house prise	Define type of a tumor, is price > 500k$
Distribution type	Normal	Binomial

Linear examples:
- Predicting the height of an adult based on the mother's and father's height
- Predicting pumpkin sales volume based on the price, time of year, and store location

Logistic examples:
- Predicting if a person will get a disease based on status, salary, genetics
- Prediction if a person will quit a job based on meetings, pull requests, office time.
- Predicting the marriage of a person based by car, salary, outlook, office time, education, country.

7. Removing correlated features

Before train model it may be worth to perform "dimensionality reduction" to save space, without loosing too much information

8. Selecting a performance measure for regression

1. Mean absolute error

$$\text{MAE} = \frac{\sum_{i-1}^{n} |e_i|}{n}$$

Disregard difference between under,over estimation.

2. Mean squared error

$$\text{MSE} = \frac{\sum_{i-1}^{n} (e_i)^2}{n}$$

3. Root mean squared error

$$\text{RMSE} = \sqrt{\frac{\sum_{i-1}^{n} (e_i)^2}{n}}$$

9. Lib

- https://www.statlect.com/
- https://towardsdatascience.com/
- https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc
- https://en.wikipedia.org/wiki/Logistic_regression