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)

$\boldsymbol{Y}$ - the variable that we predict.

Feature($\boldsymbol{x}$) 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. Supervised

Solves regression and classification tasks.

$$X \longrightarrow F \longrightarrow y$$

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

1.2. Unsupervised

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

1.3. Reinforement learning

TODO

2. Optimisation and loss function

2.1. Gradient descent

Gradient($\boldsymbol{\nabla f}$) 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.

2.2. 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 paraneter:

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

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.

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

2.3. 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.

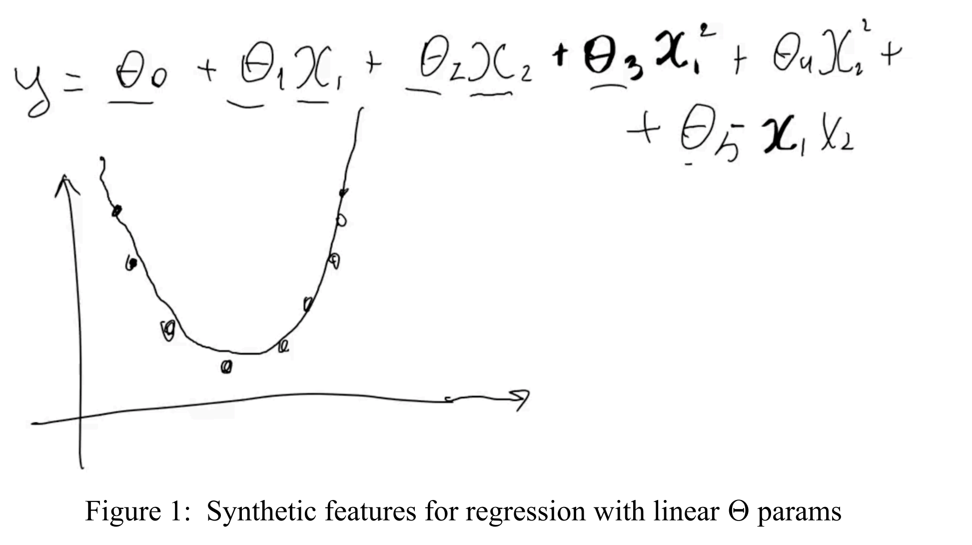

Figure 1: Synthetic features for regression with linear Θ params

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

2.3.1. Normal Linear Regression Model

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

3. 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

3.1. 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):

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

4. Classification

$y \in \{1, 2, 3, .., k\}$

Task make a function, that separates known classes.

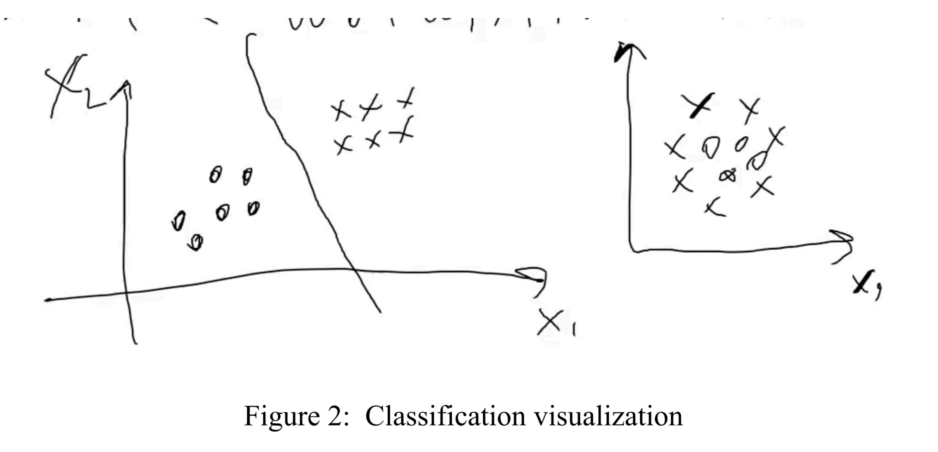

Figure 2: Classification visualization

Accordingly to image, linear regression is not suitable for this type of task(especially right)

Firstly, we will solve binary classification task$\{0, 1\}$. Model will have 1 output - probability of x is from class 1.

4.1. Logistic regression

Logistic regression type of regression that predicts a probability of an outcome given one or more independent variables. With a threshold returned probability can be mapped to a discrete value.

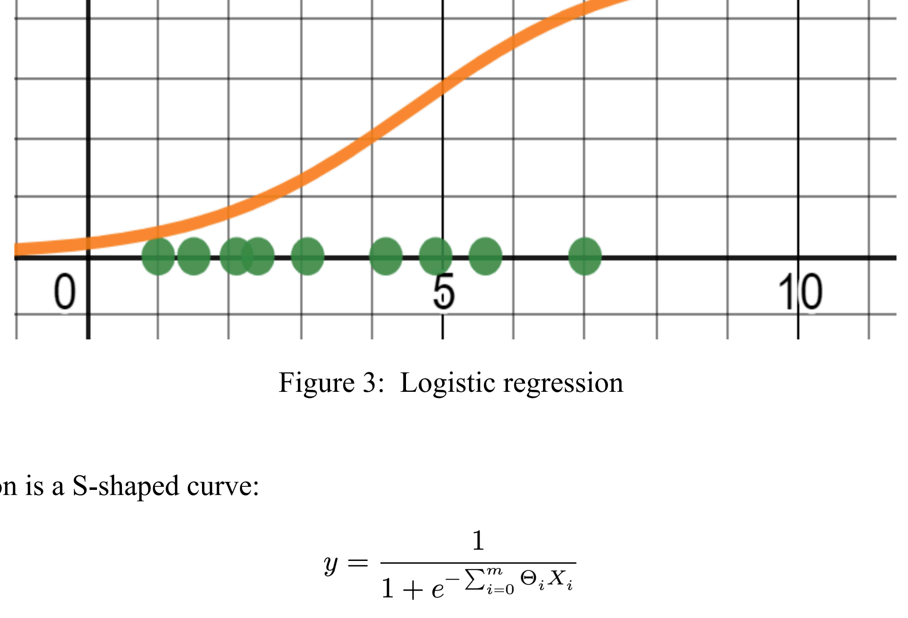

Figure 3: Logistic regression

4.1.1. Formula

Logistic regression is a S-shaped curve:

$$y = \frac{1}{1 + e^{-\sum_{i=0}^{m} \Theta_i X_i}}$$

BCE(Binary cross entropy) loss function

$$\begin{cases} -\log(p_i), y_i = 1 \\ -\log(1 - p_i), y_i = 0 \end{cases}$$

p_i - model output probability for i example. (class 1)

$$\text{BCE} = -y_i \log(p_i) - (1 - y_i) \log(1 - p_i)$$

$$\begin{cases} \text{TP } y_i, p_i = \{1, 1\} \text{ BSE} = 0 \\ \text{TN } y_i, p_i = \{0, 0\} \text{ BSE} = 0 \\ \text{FN } y_i, p_i = \{1, 0\} \text{ BSE} \to \text{inf} \\ \text{FP } y_i, p_i = \{0, 1\} \text{ BSE} \to \text{inf} \end{cases}$$

Loss for gradient:

$$L = -\frac{1}{N} \sum_{i=1}^{n} (y_i \log(p_i) + (1 - y_i) \log(1 - p_i)) \longrightarrow \min$$

$$\frac{\partial f}{\partial \Theta_j} = -\frac{1}{N} \sum_{i=1}^{n} (y_i \log(p_i) + (1 - y_i) \log(1 - p_i))'$$

4.1.2. Comparison to linear regression

4.2. Metrics

To define success of model **metrics** are used. $A = \left(\frac{N_{\text{correct}}}{N}\right) 100\%$

Precision TP/(TP+FP)
Recall TF/(TF+FN)

5. Lib

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