Deep Learning & Mathematics notes

Table des matières

[**Statistics and basic ML :** 1](#_Toc146541063)

[0-Intuitive explanation of maximum likelihood estimation 1](#_Toc146541064)

[1-why using log likelihood instead of likelihood : 1](#_Toc146541065)

[The log likelihood 1](#_Toc146541066)

[1.2 - Why to optimize max log probability instead of probability 3](#_Toc146541067)

[2-Why do we minimize the negative likelihood if it is equivalent to maximization of the likelihood? 4](#_Toc146541068)

[3. What is Bayesian inference ? 4](#_Toc146541069)

[weights initialization : 4](#_Toc146541070)

[He Weight Initialization 4](#_Toc146541071)

[Normalized Xavier Weight Initialization 5](#_Toc146541072)

[4. Bayesian Deep learning 5](#_Toc146541073)

# 

# **Statistics and basic ML :**

## 0-Intuitive explanation of maximum likelihood estimation

Maximum likelihood estimation is a method that determines values for the parameters of a model. The parameter values are found such that they maximise the likelihood that the process described by the model produced the data that were actually observed.

## 1-why using log likelihood instead of likelihood :

The logarithm of the probability of multiple joint probabilities **simplifies to the sum** of the logarithms of the individual probabilities (and the sum rule is easier than the product rule for differentiation).

## The log likelihood

The above expression for the total probability is actually quite a pain to differentiate, so it is almost always simplified by taking the natural logarithm of the expression. This is absolutely fine because the natural logarithm is a monotonically increasing function. This means that if the value on the x-axis increases, the value on the y-axis also increases (see figure below). This is important because it ensures that the maximum value of the log of the probability occurs at the same point as the original probability function. Therefore we can work with the simpler log-likelihood instead of the original likelihood.

(when we are looking for the maximum point, as long as the x doesn’t change and doesn’t change the direction it’s not a problem when multiply a function by a number, or apply a log.

* >when differentiating, we will retrieve the same x for the maximum)

Source : [Probability concepts explained: Maximum likelihood estimation | by Jonny Brooks-Bartlett | Towards Data Science](https://towardsdatascience.com/probability-concepts-explained-maximum-likelihood-estimation-c7b4342fdbb1)

Explanation :

: From a standpoint of computational complexity, you can imagine that first of all summing is less expensive than multiplication (although nowadays these are almost equal). But what is even more important, likelihoods would become very small and you will run out of your floating point precision very quickly, yielding an underflow. That's why it is way more convenient to use the logarithm of the likelihood. Simply try to calculate the likelihood by hand, using pocket calculator - almost impossible.

Une image contenant texte

Description générée automatiquement

## 1.2 - Why to optimize max log probability instead of probability

The computer uses a limited digit floating point representation of fractions, multiplying so many probabilities is guaranteed to be very very close to zero.

With log, we don't have this issue.

## 2-Why do we minimize the negative likelihood if it is equivalent to maximization of the likelihood?

optimizers in statistical packages usually work by minimizing the result of a function. If your function gives the likelihood value first it's more convenient to use logarithm in order to decrease the value returned by likelihood function. Then, since the log likelihood and likelihood function have the same increasing or decreasing trend, you can minimize the negative log likelihood in order to actually perform the maximum likelihood estimate of the function you are testing.

## 3. What is Bayesian inference ?

Now we know what Bayes’ theorem is and how to use it, we can start to answer the question what is Bayesian inference?

Firstly, (statistical) inference is the process of deducing properties about a population or probability distribution from data. We did this in my previous post on maximum likelihood. From a set of observed data points we determined the maximum likelihood estimate of the mean.

Bayesian inference is therefore just the process of deducing properties about a population or probability distribution from data using Bayes’ theorem. That’s it.

Source : [Probability concepts explained: Bayesian inference for parameter estimation. | by Jonny Brooks-Bartlett | Towards Data Science](https://towardsdatascience.com/probability-concepts-explained-bayesian-inference-for-parameter-estimation-90e8930e5348)

## weights initialization :

Classiquement, il existe différents types d’initialisation d’un modèle de deep learning :

Xavier Weight Initialization

The xavier initialization method is calculated as a random number with a uniform probability distribution (U) between the range -(1/sqrt(n)) and 1/sqrt(n), where n is the number of inputs to the node.

* weight = U [-(1/sqrt(n)), 1/sqrt(n)]

### He Weight Initialization

The he initialization method is calculated as a random number with a Gaussian probability distribution (G) with a mean of 0.0 and a standard deviation of sqrt(2/n), where n is the number of inputs to the node.

* weight = G (0.0, sqrt(2/n))

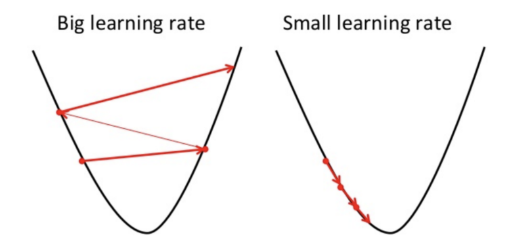
### Normalized Xavier Weight Initialization

The normalized xavier initialization method is calculated as a random number with a uniform probability distribution (U) between the range -(sqrt(6)/sqrt(n + m)) and sqrt(6)/sqrt(n + m), where n us the number of inputs to the node (e.g. number of nodes in the previous layer) and m is the number of outputs from the layer (e.g. number of nodes in the current layer).

* weight = U [-(sqrt(6)/sqrt(n + m)), sqrt(6)/sqrt(n + m)]

# Neural Networks

## Gradient Descent

* Gradient Descent is an optimization algorithm for finding a local minimum of a differentiable function. Gradient descent in machine learning is simply used to find the values of a function's parameters (coefficients) that minimize a cost function as far as possible. Why Minimizing the loss function ? It’s to minimizing the error
* A gradient measures how much the output of a function changes if you change the inputs a little bit. A gradient simply measures the change in all weights with regard to the change in error.
* How big the steps gradient descent takes into the direction of the local minimum are determined by the learning rate, which figures out how fast or slow we will move towards the optimal weights.
* Types of gradient descent:
  + Batch (Vanilla) GRADIENT DESCENT
  + Stochastic GD
  + Mini Batch GD
* ΔW = α ( -∇ LW) . W‘ = W + ΔW

## 

Source :

* [What Is Gradient Descent? | Built In](https://builtin.com/data-science/gradient-descent)

## 4. Bayesian Deep learning

Others (to do)

Some cool courses found :

1. Probas & stats basics (high school course in French) : https://lecluseo.scenari-community.org/co/AccueilTS.html

Trier quand j'aurai le temps le contenu de ces repos et les mettre ici :

- Pratique https://github.com/AmineDjeghri/nlp (mettre ça dans un dossier ici)

- https://github.com/AmineDjeghri/Reminders/blob/master/2-%20Data%20Science%20notes%20detailed.md

-