Background Review

Mustafa Hajij

Review of basic concepts in probability

Objectives:

- Review basic concepts of probability and statistics needed in deep learning.
- Review basic concepts of calculus needed in deep learning.

Probability

Probability theory forms the foundation of many concepts and algorithms in deep learning. Understanding key probability concepts is crucial for effectively working with and interpreting deep learning models. Let's explore some fundamental concepts:

Probability : Random variables

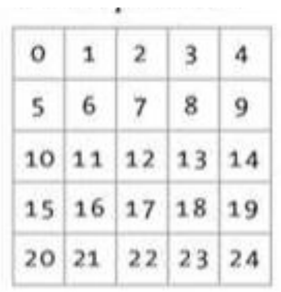
Random variables represent uncertain quantities in probability theory. In deep learning, random variables are often used to model inputs, outputs, and intermediate variables within neural networks.

Probability : Images as random variables

When working with images, each pixel can be considered a random variable. The pixel's value represents the uncertainty or variability associated with that particular location in the image. This variability can arise due to factors like lighting conditions, object appearance, noise, or variations in color or texture.

In a grayscale image, the random variables are typically discrete, taking values from 0 to 255 to represent different shades of gray. In a color image, the random variables are continuous and usually represented by the intensity levels of the red, green, and blue channels. Additionally, images can be seen as samples from an (unknown) probability distribution **P_{images}**.

Thank about the camera or whatever device that can capture images, as a way to sample from that unknown distribution.
Thank about your eyes as an instrument that can tell very well if an image is sampled from **P_{images}** or not.



0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

Probability : Images as random variables

- A probability distribution describes the likelihood of different pixel values occurring at each location in an image.
- For example, in a natural image, we often observe certain patterns and statistical properties, such as smooth color transitions, texture regularities, or object boundaries. These patterns and properties can be characterized by probability distributions.
- Deep learning models can be trained to learn these probability distributions from a large dataset of images.
- By learning the statistical patterns and relationships within the training data, the models can generate new images or make predictions on unseen images by sampling from these learned distributions (generative models).

Probability : words as random variables

By “Language as a distribution" we refer to the statistical properties of how words and sentences are arranged and used within a given language.

These statistical properties capture the probabilities of different words or sequences of words occurring in a language. In natural language processing, language is often represented as a collection of text data.

The statistical patterns and relationships within this text data can be characterized by probability distributions. For example, we can consider a language model that aims to predict the next word in a sentence given the previous context. The model learns the conditional probabilities of different words occurring given the context.

Probability : words as random variables

These probabilities form a distribution over the vocabulary, where each word has a likelihood of being the next word.

Language distributions also encompass syntactic and semantic relationships between words. For instance, the distribution can capture the probability of a particular verb being followed by a specific noun, or the probability of a word being an adjective or a verb.

Deep learning models can be trained to learn and model these language distributions from large corpora of text data.

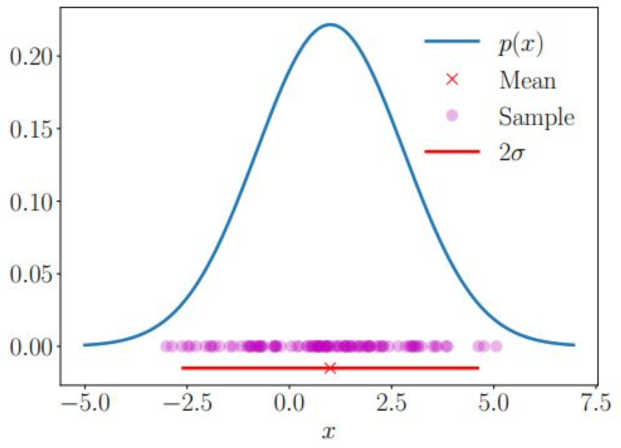
By understanding the statistical properties of language, these models can generate coherent sentences, complete text prompts, perform language translation, sentiment analysis, and various other language-related tasks.

Probability Distributions

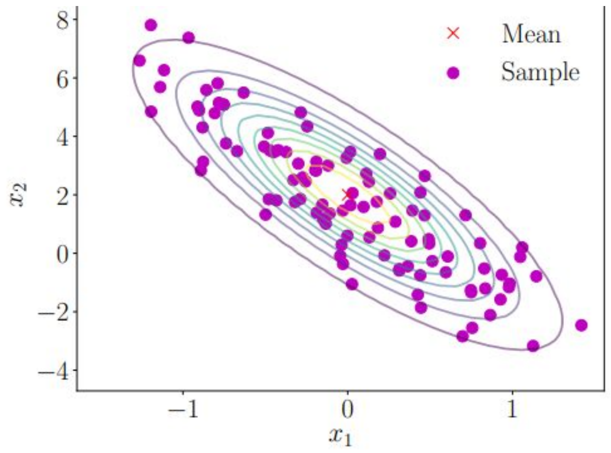
What is a PDF?

How are we given a PDF in practice ?

Why is that important to us ?



(a) Univariate (one-dimensional) Gaussian; The red cross shows the mean and the red line shows the extent of the variance.



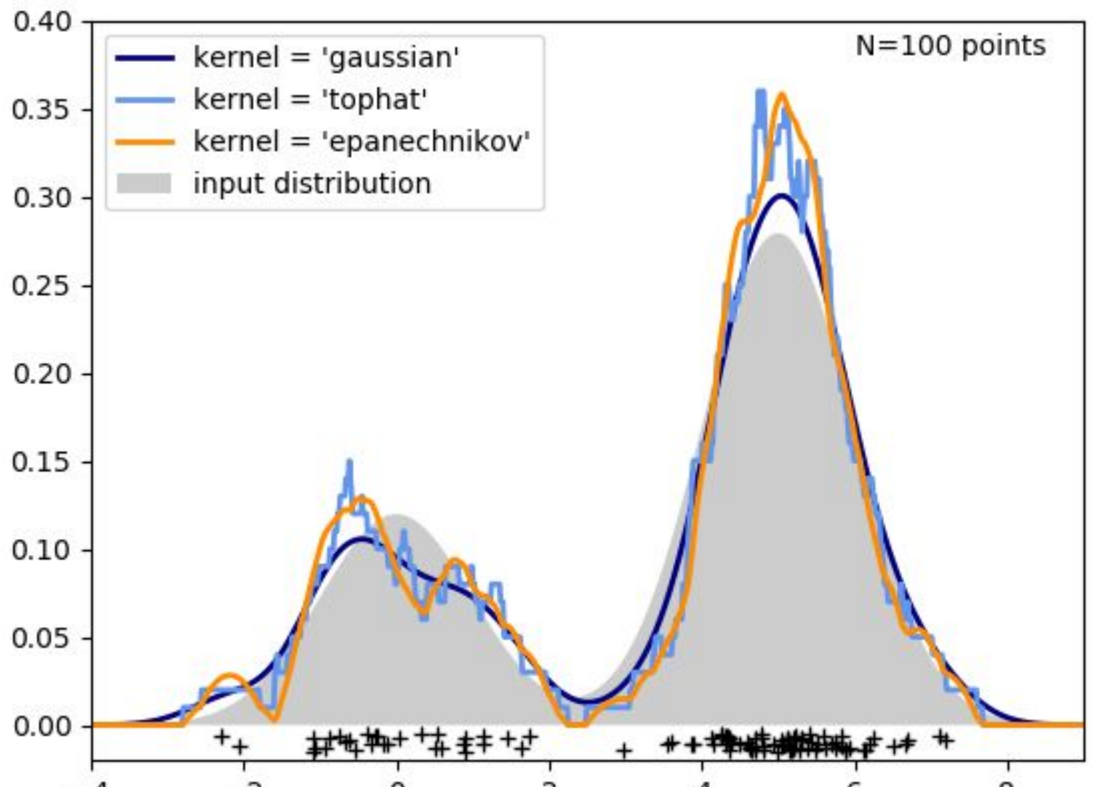
(b) Multivariate (two-dimensional) Gaussian, viewed from top. The red cross shows the mean and the colored lines show the contour lines of the density.

Image source

https://mml-book.github.io/book/mml-book.pdf

Fitting a Probability Distribution

Given enough samples from data.
How can we find its underlying PDF ?

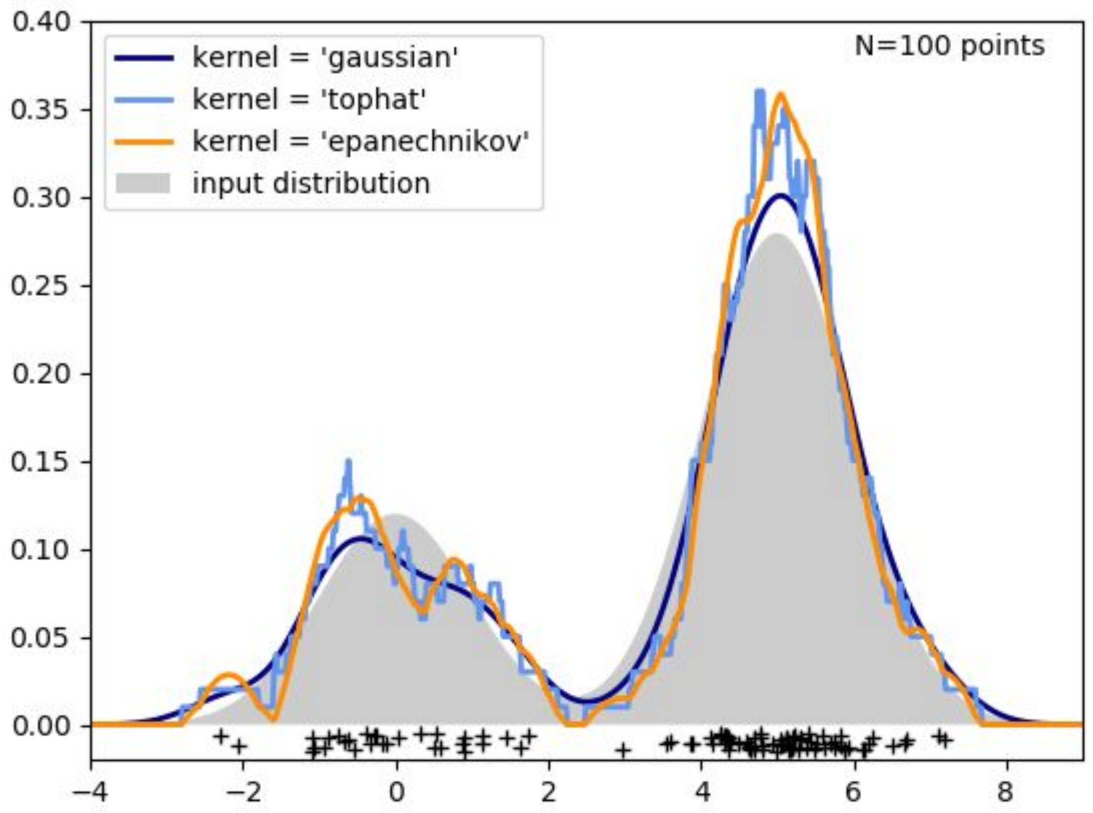


Fitting a Probability Distribution

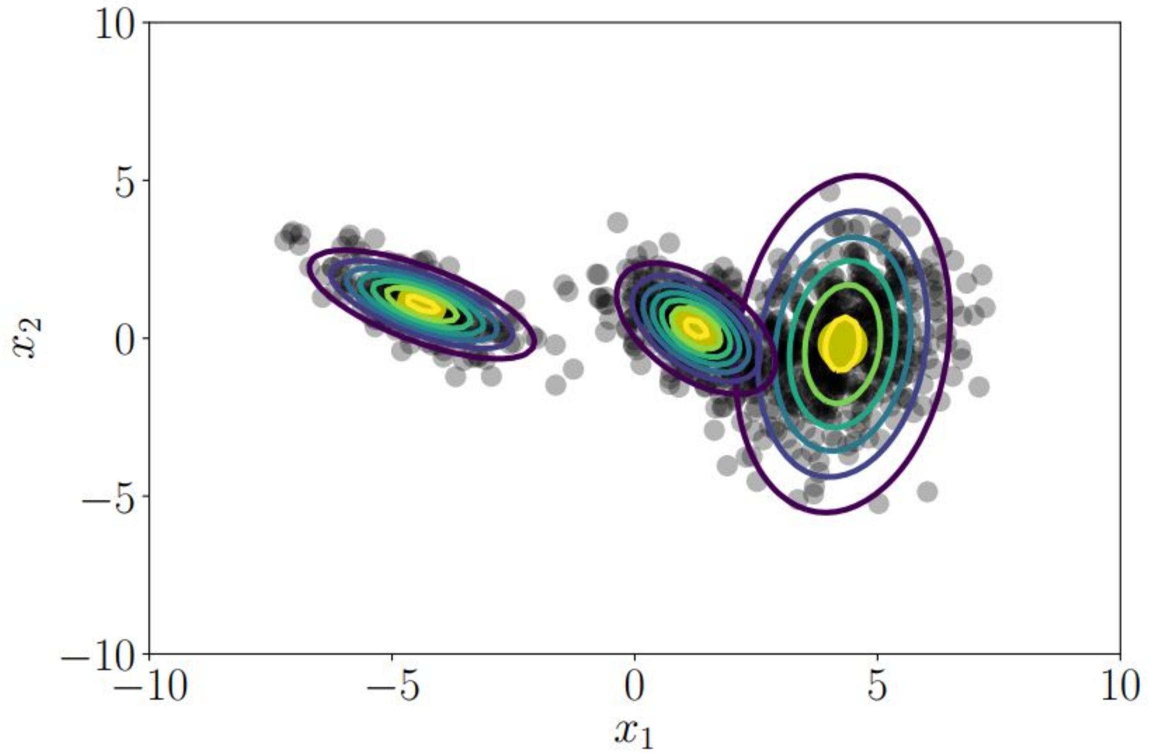
PDF is known : (a common method) Maximal Likelihood Estimation

PDF is unknown : (one of the methods) Kernel Density Estimation

$$\rho_K(y) = \sum_{i=1}^{N} K((y - x_i)/h)$$



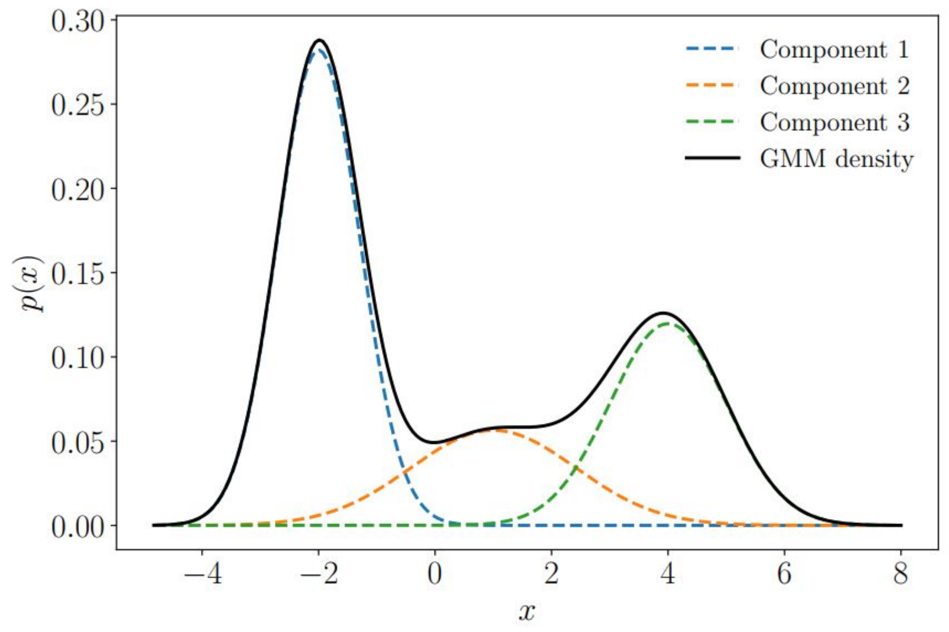
Fitting a Probability Distribution



Density estimation with a Gaussian mixture model: Find means and covariances, such that the data (dots) can be explained well.

Fitting a Probability Distribution

Fitting arbitrary complex PDF



$$p(x \mid \boldsymbol{\theta}) = 0.5\mathcal{N}\left(x \mid -2, \tfrac{1}{2}\right) + 0.2\mathcal{N}\left(x \mid 1, 2\right) + 0.3\mathcal{N}\left(x \mid 4, 1\right).$$

Image source

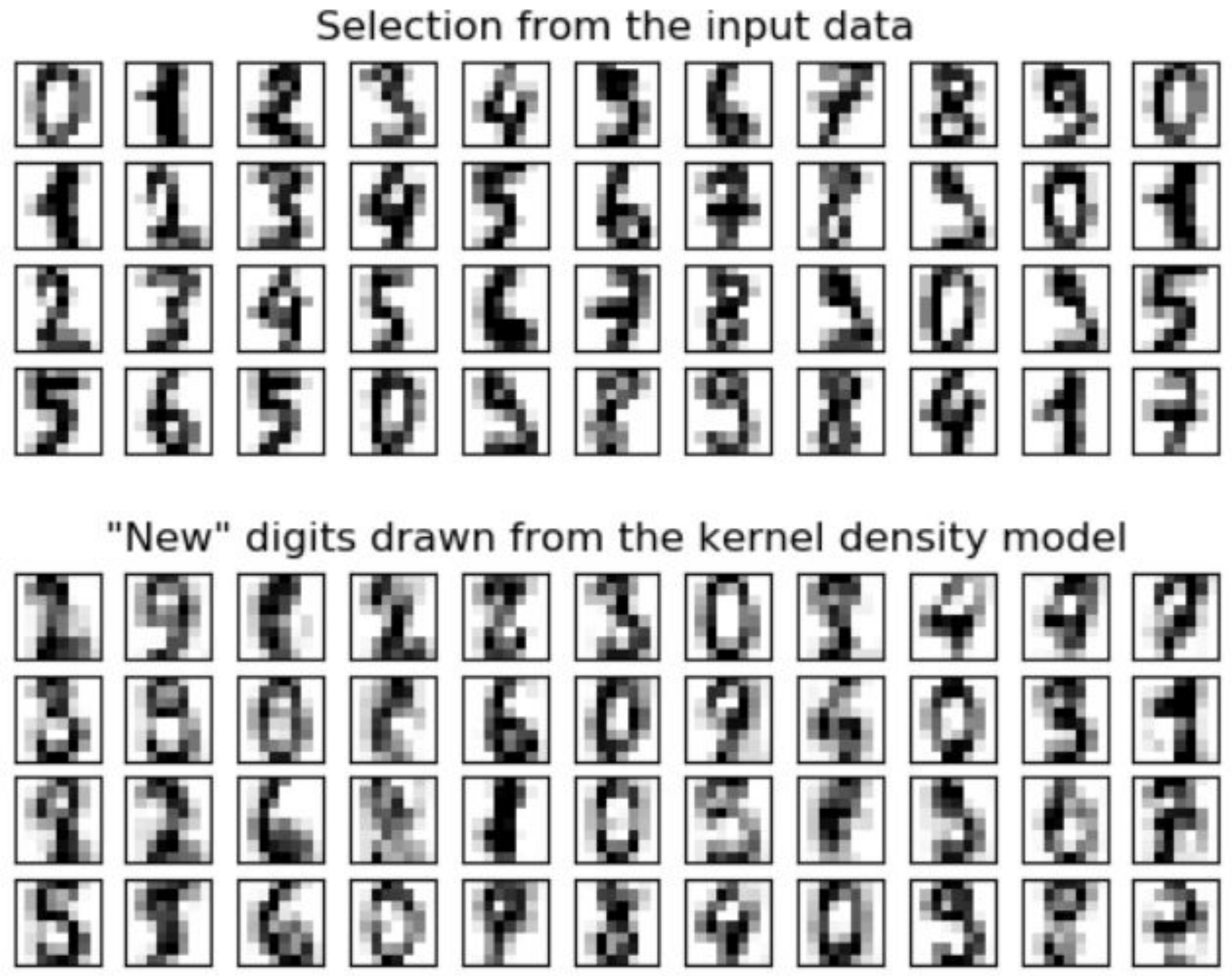
https://mml-book.github.io/book/mml-book.pdf

Sampling from PD

After fitting a PDF, some PDF can be used to generate more data samples

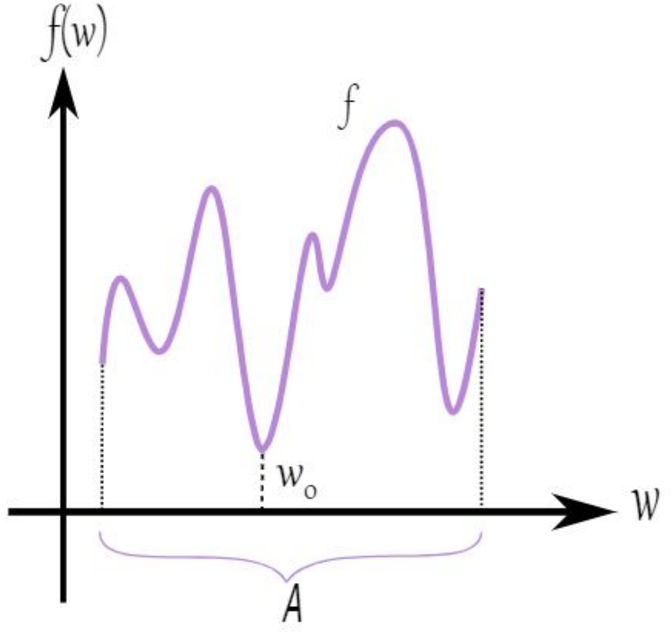
Sklearn example :

https://scikit-learn.org/stable/auto_examples/neighbors/plot_digits_kde_sampling.html

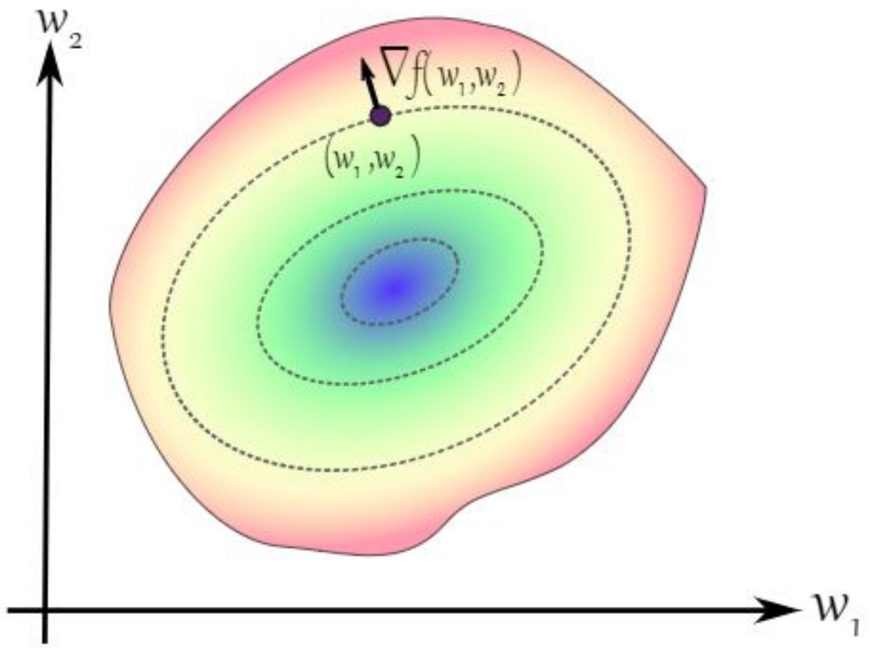

Selection from the input data
"New" digits drawn from the kernel density model

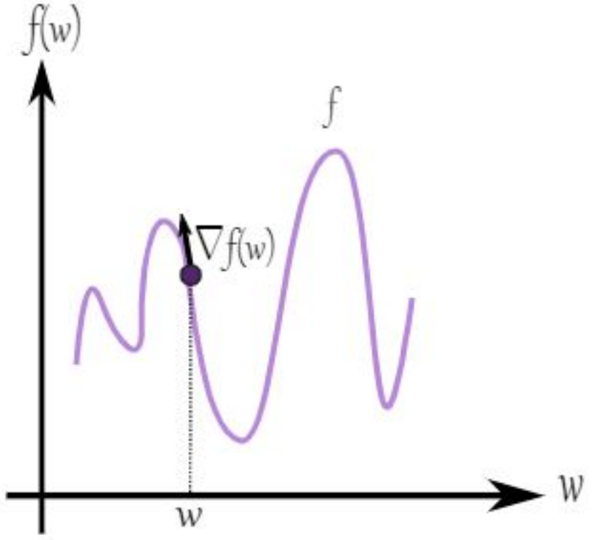
Optimization

For continuous optimization problem the problem can be generally stated as follows. For a given set A in the Euclidean space $\mathbf{R}^n$ we are giving a function $f : A \longrightarrow \mathbf{R}$, usually called the *cost or the loss function*, and the goal is to find the point $w_0 \in A$ such that $f(w_0) \leq f(w)$ for every point $w \in A$.



Differentiable functions





gradient descent

Example : Gradient Descent Algorithm

Suppose that we are given a differentiable function $f(w_1, \ldots, w_d)$

Want to find $w_1, \ldots, w_d$ such that $f(w_1, \ldots, w_d)$ is minimal. Gradient decent gives a way to find a local minimal for f

Outline :

(1) Initiate $w_1, \ldots, w_d$ randomly
(2) keep changing $w_1, \ldots, w_d$ until hopefully $f(w_1, \ldots, w_d)$ is minimal

But how exactly do we change $w_1, \ldots, w_d$?

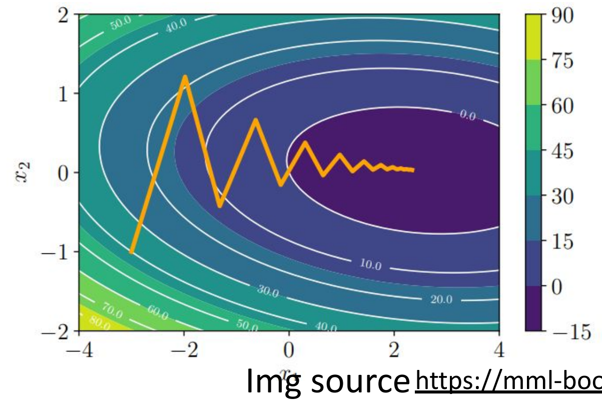
General Gradient Descent Algorithm

Key idea : gradient of f goes in the direction at which f maximally change.

```
(1) Initiate $w_1, \dots, w_d$ randomly
(2) Repeat until convergence :
    (1) For every i in range(1,d):
        (1) $w_i := w_i - q \frac{\partial f}{\partial w_i}$ (here we do simultaneous update for the parameters $w_i$ )
```

Gradient decent asserts that the values of the function f when we update as described above are non-increasing :

$$f(old\, w_i) \geq f(new\, w_i)$$

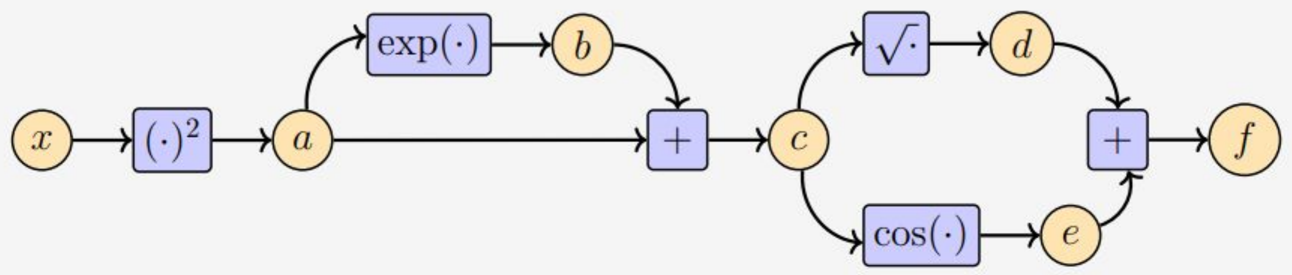


Img source https://mml-book.github.io/book/mml-book.pdf

What is a computational graph ?

$$f(x) = \sqrt{x^2 + \exp(x^2)} + \cos\left(x^2 + \exp(x^2)\right)$$

$$\begin{aligned} a &= x^2\,, \\ b &= \exp(a)\,, \\ c &= a + b\,, \\ d &= \sqrt{c}\,, \\ e &= \cos(c)\,, \\ f &= d + e\,. \end{aligned}$$



Computation graph of f

Image source

https://mml-book.github.io/book/mml-book.pdf