Gradient flows of loss functions

Le Bich Phuong

Hanoi University of Mining and Geology

Dec 19th 2018

Symmetry and singularities of gemetric structures and differential equations, Ritsumeikan University, 18-21 Dec 2018

Introduction

This talk is about some preliminary work that I'm doing under the supervision of Nguyen Tien Zung. The aim of this work is to build simple mathematical models for artificial intelligence problems, in order to study them and to propose effective solutions.

The method of **stochastic gradient flow** (of a so-called *loss function*), also known as the method of **differential learning**, is a general method very often used in machine learning. However, the obtained results are not always very good. We want to see how to improve the designs of the loss functions in order to get better results.

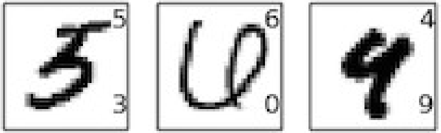
To illustrate the problems, let me begin by showing some real-world examples of machine learning, and pointing out two main kinds of difficulties in these examples, namely the **boundary cases** and the problem of **imbalanced data**.

Example 1: Recognition of hand-written numbers

A popular dataset for learning the machine learning is the MNIST database (Modified National Institute of Standards and Technology database) of 70000 grayscale images of size 28×28 pixels of hand-written digits (60000 images for training and 10000 images for testing, see, e.g.: https://en.wikipedia.org/wiki/MNIST_database). The data in this set are balanced, that is, each digit (from 0 to 9) occupies 1/10th of the set.

This is considered to be a completely solved problem: the best AI models' error rate is only 0.2%.

Nevertheless, in the *boundary cases*, for example a hand written digit which looks like a 5 and a 3 at the same time, even the best AIs can get it wrong.



Some "boundary cases" of hand-written digits: 3 or 5? 0 or 6? 4 or 9 ? ...

In general, in any classification problem, **boundary cases are difficult** and often get a wrong answer, while "interior" cases are easier to deal with.

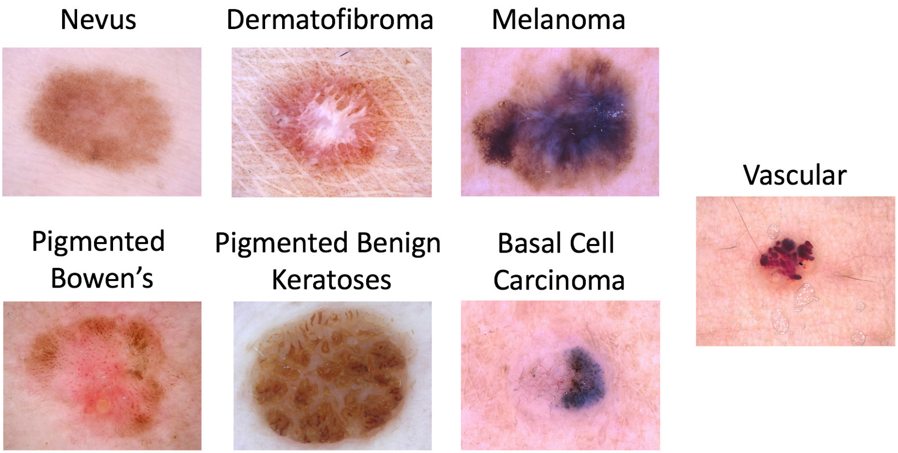
One of the main problems in machine learning is how to improve the accuracy of prediction for boundary cases.

Example 2: Skin cancer classification

The second example is the problem of classification of cancerous skin lesions into 7 types: MEL (melanoma - the most deadly type), NV (melanocytic nevus), BCC (basal cell carcinoma), AKIEC (actinic keratosis / Bowen's disease), BKL (benign keratosis), DF (dermatofibroma) and VASC (vascular lesion). The database of the International Skin Imaging Collaboration 2018 Challenge (https://challenge2018.isic-archive.com/) for this classification task has the following numbers of images for each type:

NV 6705; MEL 1113; BCC 514; AKIEC 327; BKL 1099; DF 115; VASC 142

Here we have very **imbalanced data**, which can lead to a kind of **bias** or **discrimination against minority categories** in machine learning. For example, *dermatofibroma* (DF) occupies just 1% of all data, so even if the machine cannot detect any dermatofibroma case it can still have an impressive total accuracy rate of 99%, and this makes the learning of detection of dermatofibroma difficult.



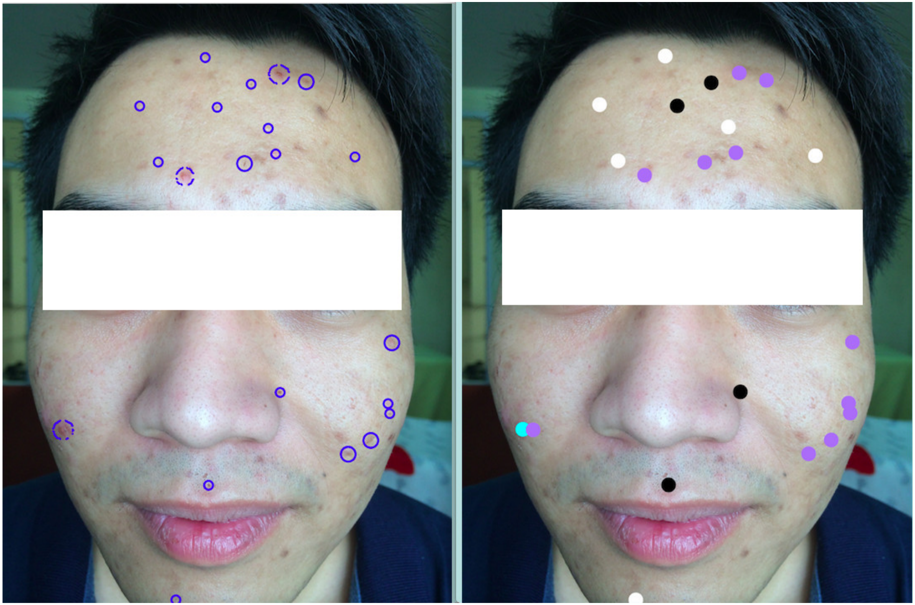
An illustration of the 7 types of cancerous skin lesions. Human doctors' diagnosis is correct only 50% of the times. Best AI's balanced accuracy rate is about 88.5% (by a company called MetaOptima, using 50000 labeled images for training. Prof. Zung's AI group's balanced accuracy rate: 87%, using 10000 publicly available labeled images for training.)

Example 3: Acne segmentation and classification

Prof. Zung's AI group was also working on an acne AI project with the following two main tasks: to segment all the acne pimples/lesions on an image, and to classify these pimples/lesions in to the following six types. The number of images in a preliminary dataset for each type is as follows:

White head: 1722; Black head: 357; Papule: 2130; Pustule: 277; Nodule: 53; Cystic: 469

Here we are faced with several difficult problems at once: **imbalanced data** (very few nodules compared to papules for example); **small objects** (acne pimples/lesions occupy just a small part of the picture) which is also a kind of imbalanced data problem; and also the **ill-defined boundary** problem (the boundary of an acne pimple/lesion is not well-defined,which makes this segmentation probelm even harder than many other segmentation problems).



An example of machine learning for acne segmentation and classification.

A general setting for differential learning (1)

Here will look only at the binary classification problems (other detection and classification problems are similar). One wants to create (e.g. by a convolutional neural network) a binary prediction machine (predictor) $\hat{M}$, i.e., a calculable binary function $\hat{M} : \Omega \to \{+1, -1\}$ (or $\{yes, no\}$) on a space Ω of all possible data of some given format (e.g., all pictures of cats and dogs, $\hat{M}$ will predict whether it's a cat or a dog). One wants $\hat{M}$ to be as close to the ground truth binary classification function

$$D : \Omega \to \{+1, -1\}$$

as possible. The machine learning way of doing it is to construct a binary map

$$M : \Theta \times \Omega \to \{+1, -1\}$$

which depends not only on Ω but also on a very large space of (*learnable*, i.e., modifiable) parameters Θ, and then learn a particular $\hat{\theta} \in \Theta$ such that $\hat{M} := M_{\hat{\theta}} = M(\hat{\theta}, .)$ is as close to the ground truth map D as possible.

A general setting for differential learning (2)

In *differential learning*, one may define

$$M_\theta(\omega) = -1 \text{ if } P(\theta,\omega) < 1/2; \quad M_\theta(\omega) = +1 \text{ if } P(\theta,\omega) \geq 1/2$$

where

$$P : \Theta \times \Omega \to [0,1]$$

is a function given by the model. The value $P_\theta(\omega) = P(\theta,\omega) \in [0,1]$ may be interpreted as the *likelihood* (probability, level of confidence), according a given parameter θ, that $D(\omega)$ will be $+1$.

One may transform P by an increasing function $\psi : [0,1] \to [-1,1]$ (or to some other interval containing 0 in the middle) such that $\psi(0.5) = 0$, call $g = \psi \circ P$ the *indicating function*, (or "*gain function*", whose opposite is called a "*loss function*"), and define $M_\theta(\omega) = sign(g(\theta,\omega))$. One then defines a **total loss function** $L : \Theta \to \mathbb{R}$ by the formula

$$L(\theta) = -\int_\Omega D(\omega) g(\theta,\omega) d\mu$$

where μ is some (empirical) probability measure on Ω.

A general setting for differential learning (3)

The idea is that elements $\omega \in \Omega$ for which the prediction by M_θ is correct, i.e. $M_\theta(\omega) = D(\omega)$, contribute negatively to the total loss function while elements ω for which the prediction by M_θ is wrong contribute positively to the total loss function. Intuitively, the smaller the loss, the better the predictor, and mimimal losses should correspond to optimal predictors. The loss function is differentiable with non-trivial differnetial (almost everywhere) with respect to $\theta \in \Theta$, and one can use a discretized stochastic gradient flow of this function on the space Θ of parameters to find a minimal point. This flow is the iterative learning process:

$$\theta_0 \mapsto \theta_1 \mapsto \theta_2 \mapsto \ldots \mapsto \theta_n \mapsto \ldots$$

where

$$\theta_{i+1} = \theta_i - \alpha \nabla L(\theta_i),$$

where α is a chosen small positive number, called the **learning rate**, and ∇L denotes the gradient of L. Hopefully, for some n big enough, M_{θ_n} is a good approximation of D, with a low level of error.

Problems with the gradient flow (1)

In practice, the above differential learning method works well, modulo some modifications in order to address the following issues:

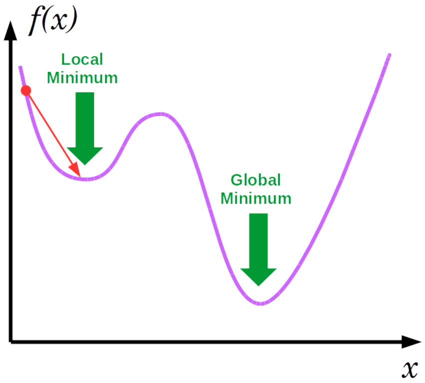
1) It's impossible in practice to compute the loss L and its gradient ∇L exactly. (In a deep learning model, Θ will have millions of dimensions, and the exact computation of ∇L would require too many operations).

Solution: Calculate only a random partial gradient, and as a result, we get a **stochastic gradient flow**.

2) The gradient flow (starting from some initial value) may get stuck at a bad local minimum or saddle point instead of going to a global minimum.

Solution: Add momemtum and noise in order to jump out of local minima and saddle points. So we get a **noisy stochastic random gradient flow with momentum** (which is almost the same as a *damped noisy stochastic Hamiltonian flow*: gradient means damped, momemtum means Hamiltonian)

Problems with the gradient flow (2)



3) The loss function, if not well chosen, may be a bad proxy for the inaccuracy level: it may happen that minima of the loss functions correspond to mediocre predictors and not to the accurate ones.

Solution: Choose an appropriate empirical probability measure on Ω to "rebalance" data, and an appropriate design of the loss function?

A very simple model (1)

In order to understand this issue 3, we will study a very simple toy model: the space Ω of all possible inputs is just an interval:

$$\Omega = [a, b[$$

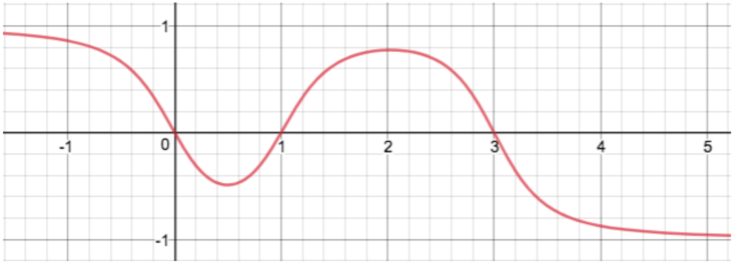
The ground truth binary classification function is piece-wise constant, i.e., there is a partition of Ω into a finite number of intervals, $\Omega = \bigcup_{i=0}^{n} [a_i, a_{i+1}[$ with $a = a_0 < a_1 < \cdots < a_{n+1} = b$, and D is equal to $+1$ on $\Omega_+ = \bigcup [a_{2i}, a_{2i+1}[$ and to -1 on $\Omega_- = \bigcup [a_{2i+1}, a_{2i+2}[$.

The *gain function* g (opposite to the loss function) has n learnable parameters $\theta_1, \ldots, \theta_n$ and is of the type

$$g(\theta_1, \ldots, \theta_n, \omega) = \prod_{i=1}^{n} \phi(\omega - \theta_i)$$

where $\phi(x)$ is a decreasing monotonous function on $\mathbb{R}$ such that $\phi(0) = 0$, and $\lim_{x \to -\infty} \phi(x) = 1$, $\lim_{x \to +\infty} \phi(x) = -1$. For example, we can take $\phi(x) = \frac{-x}{\sqrt{x^2 + \epsilon}}$ (or $\phi(x) = -\frac{2}{\pi} \arctan(x/\epsilon)$) for some positive number ϵ, which may be called the *sharpness coefficient*.

A very simple model (2)



An example of g with $\theta = (\theta_1, \theta_2, \theta_3) = (0, 1, 3)$ and $\phi(x) = \frac{-x}{\sqrt{x^2+1/4}}$

Notice that the function $g(a_1, \dots, a_n, .)$ is positive on Ω_+ and negative on Ω_-. The binary predictor

$$M_\theta(\omega) = sign(g(\theta, \omega)),$$

where $\theta = (\theta_1, \dots, \theta_n)$. So if $\theta = (\theta_1, \dots, \theta_n) = (a_1, \dots, a_n)$ then the predictor coincides with the ground truth classification function, and we get 100% accuracy.

A very simple model (3)

We do not know the value of $a_1, \ldots, a_n$, and we want to find them by the differential learning method, i.e. using the stochastic gradient flow of the gain function

$$G(\theta) = \int_a^b \left(\prod_i sign(a_i - \omega)\right) g(\theta, \omega) d\mu_\omega$$

where $d\mu_\omega$ is some measure on $[a, b]$ (by default, put $d\mu_\omega = d\omega$).

Unfortunately, in general the maximal value of $G(\theta)$ is not at the point $\theta = (a_1, \ldots, a_n)$ in the parameter space, but at some nearby point at best.

In other words, in general, the differential learning method does not give a predictor with 100% accuracy even when such a predictor exists.

A very simple model (4)

For simplicity, consider the case with just one scalar parameter: $n = 1$, $g(\theta, \omega) = \phi(\omega - \theta)$, and $d\mu_\omega = d\omega$. It is easy to see that in this case we have:

$$\frac{dG(\theta)}{d\theta} = g(\theta, b) + g(\theta, a) - 2g(\theta, a_1) = \phi(b - \theta) + \phi(a - \theta) - 2\phi(a_1 - \theta)$$

Balanced situation. (Symmetry between the positive and the negative parts), when $b - a_1 = a_1 - a$, i.e. , $a_1 = (a + b)/2$, we have $\frac{dG(\theta)}{d\theta} = 0$ at $\theta = a_1$, and a_1 is the maximal point for $G(\theta)$, and so in principle the gradient flow will converge to this good value.

Unbalanced situation. When $b - a_1 > a_1 - a$ (but $|b + a - 2a_1|$ is small enough), then $\frac{dG(\theta)}{d\theta}$ vanishes at a point in the interval $[a, a_1[$ near a_1 and not at a_1. Indeed, if $b - a_1 > a_1 - a$ (more "no" than "yes" in the data), then $\frac{dG(\theta)}{d\theta}(a_1) < 0$, which implies that the argmax of G is on the left of a_1.

A very simple model (5)

In other words, even if the stochastic gradient flow of G converges to this argmax point, it does not give the best predictor in the family: this argmax predictor is biased against the minority "yes", and there will be fewer "yes" in the prediction than in reality.

Very imbalanced situation rare event situation. If $b - a_1 >> a_1 - a$ so that $\phi(a-b) \geq 2\phi(a-a_1)$, then the derivative of G is always negative on $[a, b]$, which means that argmax of G on $[a, b]$ is a. In other words, the gradient flow will converge to a. The corresponding predictor predicts every situation as a "no" (i.e. -1), and no situation as a "yes" (i.e. $+1$), the rare cases "yes" are completely ignored

The case with $n \geq 2$ parameters is similar: the argmax does not correspond to the best predictor, i.e. it does not coincide with $(a_1, \ldots, a_n)$ in general.

The above situations really happen in practice: the imbalance of data makes the learning difficult, and there is a bias/discrimination against the minorities (unless these minorities can be "augmented" somehow).

Fighting the bias (1)

Two main ways of fighting discrimination (bias against minorities, i.e. small categories in imbalanced data) and improving the accuracy of prediction:

1) Use sharper gain/loss functions.

2) Augment the minorities.

For example, in the formula

$$\phi(x) = \frac{-x}{\sqrt{x^2 + \epsilon}},$$

when ϵ is very small, then ϕ is "very sharp" and the minimal point of the gain function G in our simple model is very near the optimal value $(a_1, \ldots, a_n)$ for the predictor, even when the data is imbalanced.

There is, however, a price to pay for the sharpness: the stochastic gradient flow will become more noisy/stochastic when ϵ becomes smaller, which leads to a stochastic equilibrium which is further away from the minimal point.

Fighting the bias (2): sharper loss functions

Indeed, ϵ small means the derivative of $\phi(x) = \frac{-x}{\sqrt{x^2+\epsilon}}$ is high near $x = 0$. The empirical stochastic gradient is computed over a relatively small sample (called a *batch*) in general, and so its value can be very far from the true gradient value. It means that when ϵ is small then the variance in the stochastic gradient flow is high.

By general results on stochastic dynamical systems, we know that when the variance is high then the stochastic equilibrium (to which the flow converges) is far away from the minimal point of the loss function.

In the limit case, when $\epsilon = 0$, the emperical loss function becomes piecewise constant, its deriviative is zero almost everywhere and infinite at some places, and the gradient descent method does not work at all.

Idea: sharpen the loss function but decrease the learning rate at the same time.

Fighting the bias (3): augmentation of minorities

Rebalancing by augmentation: Take a measure distribution on Ω which gives more weight to each minority element than to a majority element.

Proposition: *In our simple toy model with n parameters, for any $a = a_0 < a_1 < \ldots < a_{n+1} = b$ there exists a (unique probability) measure $d\mu = p d\omega$ with p positive constant on each interval $]a_i, a_{i+1}]$ such that the argmax of the gain function $G(\theta)$ is exactly $(a_1, \ldots, a_n)$.*

The "rebalancing by augmentation" idea works well in practice. But what weights to choose? Pick them by hand ?!

It is a delicate problem: if the minority is augmented by the ratio equal to (majority mass / minority mass), then instead of having a discrimination against the minority we will actually have a discrimination against the majority!

Automatic rebalancing: The idea is that the best relative weights to give to the minorities can also be found automatically by machine learning!

Fighting the bias (4): an example of augmentation

Example: Skin cancer classification. Augmentation coefficients which seem to work very well:

Type	Number of images	Coefficient
NV	6705	1
MEL	1113	4
BCC	514	8
AKIEC	327	10
BKL	1099	4
DF	115	25
VASC	142	22

Note that
$6705 \times 1 > 1113 \times 4 > 1199 \times 4 > 514 \times 8 > 317 \times 10 > 142 \times 22 > 115 \times 25$.
The augmentation coefficients are integers because they are easy to implement in machine learning. If say the coefficient is 2 then feed the same thing twice to the data pipe (with some image transformations).

THANK YOU FOR YOUR ATTENTION!