Topics:

- …

How measure errors. Terminology.

1. *Quanity* in which you interesting in and try to find via some approach. Let's say it's **x**
2. Real best ***Qunaity*** that you should obtain is **x***. Let's assume it's unique
3. You don't know it because it's true answer and not known
4. Really you can not measure **|x-x*|** because you don't know **x*.** If you know **x*** there is no real reason to setup such problem at all.
5. What you can measure is *Quanity* **|f(x)-f(x*)|**
6. Quanity **|x-x*|** in English literature called *backward error*
7. Quanity **|f(x)-f(x*)|** in English literature t's called *forward error*

Meaning of terminology

https://youtu.be/Ib18ONxvz2Y?list=PLrX-Xj1yIkS-PcenrKFF3XocjxayjPgHE&t=229

03:55

“Имена не дают знаний, это только какие-то имена. Что мой отец забыл мне сказать, что имена нужны если ты хочешь с кем-то поговорить” – Ричард Фейнман

Forward and backward error

Everything from previous slides are names.

Real meaning

$|f(x)-f(x^*)|$ can be (typically) measured

$|x-x^*|$ can not be (typically) measured.

Absolute error and relative error

- math.abs(1-3*(4/3.0-1)) is absolute Error
- math.abs(1-3*(4/3.0-1))/1.0 is relative Error

Hint

You can use this expression to evaluate what is a typical **error** in **computations** with real number in your computer or programming language

What we can do ?

If Aggregate different areas then there are 5 ways:

A. Nothing. Just stop algorithm when **|f(x)-f(x*)| < tolerance**. It is some criteria, but really alone it say nothing. Really we hope that when "**f(x)-f(x*)**" is small then **x is near x***
It's in fact definition of well-condition problems "if small forward error implies small backward error" then problem is well-conditioned.

B. Sometimes from forward error you can obtain bound in backward error.

C. Sometimes you can derive bound on **x-x*** a solution which depends on uknown constants. Carefull analyze should be done how this numbers can be replaced with exact numbers.

D. It's more lile exception, but amazingly sometimes people derive different bound which <u>does not depend</u> in any unknown constants.

E. If problem can be solved exactly in integer arithmetic then everything then typically the solution is exact

Subtle thing around unconstrained

Really there are more sublte things during optimization
$a = agrmin_a \boldsymbol{s}(\boldsymbol{a})$

1. Your step can lead to place where function is not defined
2. Algorithm can diverge (разойтись)
3. Algorithm can converge (сойтись) very slowly such that algorithms is **inpractical**
4. Maybe function does not have gradient
5. You should understand stop criteria
6. For differentiable function $|\nabla f|_2^2 \le eps$ is good heuristic stop criteria. But it’s heuristic.
7. For non-heuristic more correct rules should be derived

Variants of gradient descend algorithms

- Batched steepest-descend
- Online steepest-descend
- Iterative online steepest-descend with momentum

Fast First order methods

[1] **Heavy Ball (Polyak) method**
$x^{k+1} = x^k - a_k g^k + \beta_k(x^k - x^{k-1}), a_k > 0, \beta_k > 0$
The role of second term is momentum.

[2] **Momentum**
$s^k = (1-\beta)g^k + \beta s^{k-1}$. "**Momentum in ML**"
(in fact it is **filtered (sub)gradient method**)

[3] **Nesterov Momentum**
$s^k = (1-\beta)\nabla f(x^k + \beta s^{k-1}) + \beta s^{k-1}$
Instead g^k we incorporate information about gradient from place where we will land via consider $x^k + \beta s^{k-1}$:

[4] **AdaGrad**
$\Sigma^k = (\mathrm{g}^1)^2 + (\mathrm{g}^2)^2 + \cdots (\mathrm{g}^k)^2$ – running sum of squares of gradients over all steps.

$s^k = \frac{\mathrm{g}^k}{(\sqrt{\Sigma^k} + eps)}$
(John Duchi who is this day professor in Electrical Engineering at Stanford worked on it during doing his PhD.)

Fast First order methods

[5] RmsProp

$$\Sigma^k = decay(\Sigma^{k-1}) + (1 - decay)(g^k)^2$$
$$s^k = \frac{g^k}{(\sqrt{\Sigma^k} + eps)}$$

[6] **Adam**. Adaptive Moment Estimation.

- $m_1 = decay(m_{1,prev}) + (1 - decay)g^k$ (Momentum for g^k)
- $m_2 = decay(m_{2,prev}) + (1 - decay)(g^k)^2$ (Momentum for $(g^k)^2$)
- $m_1^{unbias} = \frac{m_1}{(1-\beta_1{}^t)}$ (Bias correction)
- $m_2^{unbias} = \frac{m_2}{(1-\beta_2{}^t)}$ (Bias correction)
- $s^k = \frac{m_1^{unbias}}{(\sqrt{m_2^{unbias}} + eps)}$
- This algorithm in fact combine of Momentum and RMSProp.

Regularization Strategies

- **Early stopping.** Dividing learn set into learn/dev set (90%/10%). Idea to stop when validation goes up.
- **Drop out.** During forward and backward propagate half of the network nodes *output* is setuped to zero
- **Drop connect.** During forward and backward some *weights* are setuped to zero and just update anothers.
- **Extra regularization with extra term**
  L2 norm square in objective with all parameters

Pros and cons of Single Layer NN

Pros

- Single layer neural net is general in terms that they can approximate continuous function via appending extra units in hidden layer.
- Theorem said that you should have no deep layers if number of data is infinite, but it's not infinite.
- Result weights depends on anything because arised optimization problem is non-convex.
- Applied to many complex problems
- It's possible to be employed for work on this field. Create big market for predictive models

.

Cons

- Many tunable parameters: topology of network, gradient momentum, learning rate, type of transfer function, and etc.
- Training very slow.
- Produce black-box models which are not easy to intepretate even people working on it right now.
- Degree of success depend on skill of engineer, they are not out-of-shelve.