

Recall the situation, the Deep, feed forward neural net (also called MLP = multi layered perceptron)

has the form

$$F(x, \eta) = F_L \circ \dots \circ F_1(x)$$

$$F_i(x) = \sigma(A_i x + \vec{b}_i)$$

The parameters are $\eta = A_1, \dots, A_L, b_1, \dots, b_L$ with correct output

The training data is x_1, \dots, x_N with corresponding y_1, y_2, \dots, y_N function is (simplest version)

The loss or error or objective function is

$$\Phi(\eta) = \frac{1}{N} \sum_{i=1}^N \|F(x_i, \eta) - y_i\|^2$$

Learning consists of adjusting the parameters η so that the error diminishes.

• To diminish Φ the first step is to find the direction we should move from \vec{x} so that Φ gets smaller. We had to next theorem from calculus.

Summary: Given $\Phi: \mathbb{R}^n \rightarrow \mathbb{R}$ at a point $\vec{x} \in \mathbb{R}^n$

the direction of maximal increase of Φ is given by

$\nabla \Phi(\vec{x})$ and the direction of maximal decrease is

$$-\nabla \Phi(\vec{x}).$$

Further, for a level

given by $\vec{x} \in \mathbb{R}^n: \Phi(\vec{x}) = c$ at a point

set $L_c = \{ \vec{x} \in \mathbb{R}^n: \Phi(\vec{x}) = c \}$ is perpendicular to

$\vec{x} \in L_c$, $\nabla \Phi(\vec{x})$ is perpendicular to

the level set [provided level set is nice

at x , no corners, etc.]

The idea of gradient descent is to take a step of size h in the direction in which Φ is decreasing the most, i.e. in the direction $-\nabla\Phi(x)$

OR $x_0 = \text{initial position}$

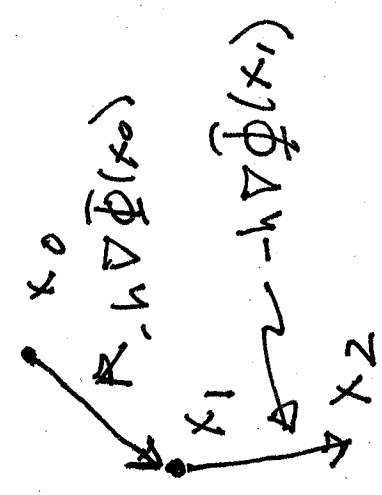
$$x_1 = x_0 + h(-\nabla\Phi(x_0))$$

we repeat

$$x_2 = x_1 + h(-\nabla\Phi(x_1))$$

⋮

or as a loop



- Input: x_0 and stepsize h and stop criteria
- Requires: subroutine to compute $\nabla\Phi(x)$

• For $i = 1$ to n

$$x_i = x_{i-1} - h \nabla\Phi(x_{i-1})$$

end

- Two questions
- how do we choose h ?
- when do we halt?

Halting is a bit more straight forward at first

Standard things are

- $|x_{i+1} - x_i| < \epsilon$, so the iteration isn't making much progress. Note that since $-\hbar \nabla \Phi(x_{i-1})$

$x_L = x_{L-1}$ halting when

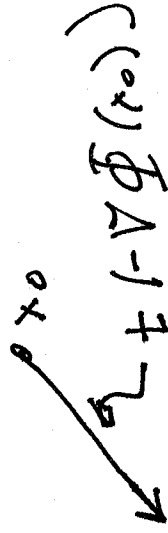
This is equivalent to $\|\nabla \bar{\Phi}(x_i)\|$ is small

- $|\Phi(x_{i+1}) - \Phi(x_i)| < \epsilon$, so the function values aren't changing much

- BUT we could be in a gradually sloping valley or at a local min ...

Picking h (also called the learning rate) has many aspects.

- The first idea is to do a line search

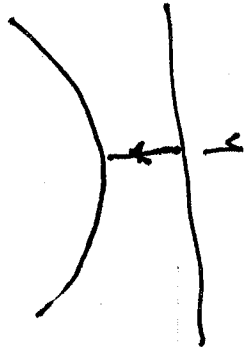


A diagram showing a point labeled x_0 at the end of a vector. From the tip of this vector, another vector points downwards and to the right, labeled $t(-\nabla\Phi(x_0))$.

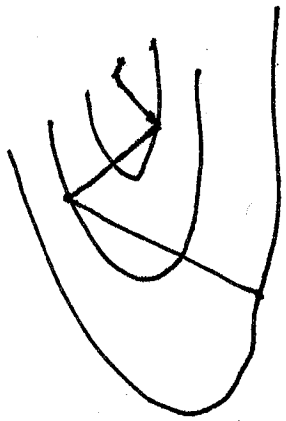
and then

let $g(t) = \Phi(x_0 - t \nabla\Phi(x_0))$ as a function

use a 1-D routine to minimize g and let $h = \text{argmin}(g(t))$ of t



- Even with line search gradient descent can get caught in "zig-zags".



(see wikipedia figures)

Another idea is to choose different

directions

• h_1 for different directions
 $x_1' = x_1 - h_1 \frac{\partial \Phi}{\partial x_1}(x)$

where $x_L \rightarrow x_L'$ is
 reupdate.

$$x_2' = x_2 - h_2 \frac{\partial \Phi}{\partial x_2}(x)$$

• yet another to choose directions other than $\nabla \Phi$ (conjugate gradients)

There are many other ways to find $\min \Phi$
Optimization Theory

In the vast literature of Φ and $\nabla \Phi$ (The Hessian $H(\Phi)$)

Some methods using higher derivatives (The Hessian $H(\Phi)$)
or more evaluation of Φ and $\nabla \Phi$

in a Deep learning net

But since each evaluation of Φ and $\nabla \Phi$ is very
computationally expensive so some
variant of Gradient Descent is used

In the next lecture we will discuss
a special method to compute $\nabla \Phi$

• One last variant of Gradient Descent is important /10
to mention

• Recall we have training set x_1, \dots, x_N with
correct output y_1, \dots, y_N where $N = 10,000$ is possible
function is

• The net work's input-output function is

$$F(x, \eta) = F_L \circ \dots \circ F_1(x)$$

where each $F_L = \sigma(A_L \vec{x} + b_L)$ and the
parameters are $\eta = (A_{11}, A_{12}, b_{11}, b_{12})$

Parameters are η have a loss or error function
and we

$$\Phi(\eta) = \frac{1}{N} \sum_{L=1}^N \|F(x_i, \eta) - y_i\|^2$$

44

• Notice that one evaluation of Φ requires N evaluations of F not to say how expensive computing $\nabla \Phi$

• Stochastic Gradient Descent takes advantage of the fact that Φ is a sum of errors, one for each x_L from the training set

• This proceeds by picking a minibatch from the training data $\{x_1, \dots, x_K\}$ and then

$$\text{compute } \bar{\Phi}(\eta) = \frac{1}{K} \sum_{i=1}^K \|F(\bar{x}_L, \eta) - \bar{y}_L\|^2$$

and using this to compute $\nabla \bar{\Phi}$ and using

Gradient Descent to update η .

• Then repeat with a new, randomly chosen minibatch of data.