Part 7.5

Stochastic Gradient Descent and Stochastic Newton
**Background**

In many practical applications, the objective function is a large sum:

\[ f(x) = \sum_{i=1}^{N} f_i(x) \]

Issues and questions:

- Evaluating gradients/Hessians is expensive
- Do all of these \( f_i \) really provide *complementary* information?
- Can we exploit the sum structure somehow to make the algorithm cheaper?
Stochastic gradient descent

Approach: Let’s use gradient descent (steepest descent), but instead of using the full gradient

\[ p_k = -\alpha_k g_k = -\alpha_k \nabla f(x_k) \]

Try to approximate it somehow in each step, using only a subset of the functions \( f_i \):

\[ p_k = -\alpha_k \tilde{g}_k \]

Note: In many practical applications, the step lengths are chosen a priori, based on knowledge of the application.
**Stochastic gradient descent**

**Idea 1:** Use only one $f_i$ at a time when evaluating the gradient:

- In iteration 1, approximate
  \[
g_1 = \nabla f(x_1) \approx \nabla f_1(x_1) =: \tilde{g}_1
\]

- In iteration 2, approximate
  \[
g_2 = \nabla f(x_2) \approx \nabla f_2(x_2) =: \tilde{g}_2
\]

- ...  

- After iteration $N$, start over:\n  \[
g_{N+1} = \nabla f(x_{N+1}) \approx \nabla f_1(x_{N+1}) =: \tilde{g}_{N+1}
\]
**Stochastic gradient descent**

**Idea 2:** Use only one $f_i$ at a time, randomly chosen:

- In iteration 1, approximate

  $$g_1 = \nabla f(x_1) \approx \nabla f_{r_1}(x_1) =: \tilde{g}_1$$

- In iteration 2, approximate

  $$g_2 = \nabla f(x_2) \approx \nabla f_{r_2}(x_2) =: \tilde{g}_2$$

- ...

Here, $r_i$ are randomly chosen numbers between 1 and $N$. 
**Stochastic gradient descent**

Idea 3: Use a subset of the $f_i$ at a time, randomly chosen:

- In iteration 1, approximate

  $$g_1 = \nabla f(x_1) \approx \sum_{i \in S_1} \nabla f_i(x_1) =: \tilde{g}_1$$

- In iteration 2, approximate

  $$g_2 = \nabla f(x_2) \approx \sum_{i \in S_2} \nabla f_i(x_2) =: \tilde{g}_2$$

- ...

Here, $S_i$ are randomly chosen subsets of \{1...N\} of a fixed size, but relatively small size $M<<N$. 
Stochastic gradient descent

**Analysis:** Why might anything like this work at all?
- The approximate gradient direction in each step is wrong.
- The search direction might not even be a descent direction.
- The sum of each block of $N$ partial gradients equals one exact gradient, so there does not seem to be any savings

**But:**
- *On average*, the search direction will be correct.
- In many practical cases, the functions $f_i$ are not truly independent, but have redundancy.

**Consequence:** Far fewer than $N$ steps are necessary compared to one exact gradient step!
**Stochastic Newton**

**Idea:** The same principle can be applied for Newton’s method.

Either select a single $f$ in each iteration and approximate

\[
g_k = \nabla f(x_k) \approx \nabla f_{r_k}(x_k) =: \tilde{g}_k
\]

\[
H_k = \nabla^2 f(x_k) \approx \nabla^2 f_{r_k}(x_k) =: \tilde{H}_k
\]

Or use a small subset:

\[
g_k = \nabla f(x_k) \approx \sum_{i \in S_k} \nabla f_i(x_k) =: \tilde{g}_k
\]

\[
H_k = \nabla^2 f(x_k) \approx \sum_{i \in S_k} \nabla^2 f_i(x_k) =: \tilde{H}_k
\]
**Summary**

**Redundancy:** In many practical cases, the functions $f_i$ are not truly independent, but have redundancy.

**Stochastic methods:**

- Exploit this by only evaluating a small subset of these functions in each iteration.

- Can be shown to converge under certain conditions

- Are often faster than the original method because
  - they require vastly fewer function evaluations in each iteration
  - even though they require more iterations