Unit 19 – Gradient/Steepest Descent

Steepest Descent
- Given a cost function \( F(\vec{c}) \), recall from Unit 9 (Basic Optimization) that \( \vec{F}(\vec{c}) \), the vector of all first derivatives, is also known as the gradient and denoted \( \nabla F(\vec{c}) \)
- For any point \( \vec{c}_0 \), \( \nabla F(\vec{c}_0) \) is the direction in which \( F(\vec{c}) \) increases the fastest, and \( -\nabla F(\vec{c}_0) \) is the direction in which \( F(\vec{c}) \) decreases the fastest
- Thus, \( -\nabla F(\vec{c}_0) \) is considered the direction of steepest descent
- Using \( -\nabla F(\vec{c}_0) \) as the search direction is known as gradient/steepest descent
- This can be thought of as always “walking in” the steepest downhill direction
- But of course, never going uphill can lead to local minima

Quadratic Forms
- Given a square matrix \( A \), the quadratic form is \( f(x) = \frac{1}{2} x^T A x - b^T x + c \)
- Minimize \( f(x) \) by taking the gradient and setting it equal to zero
  - \( \nabla f(x) = \frac{1}{2} A x + \frac{1}{2} A^T x - b = A x - b = 0 \) assuming \( A \) is symmetric
  - Thus, minimizing the quadratic form given by \( f(x) \) and solving \( A x = b \) are equivalent
- The steepest descent direction for the minimization is \( -\nabla f(\vec{x}) = b - A \vec{x} = \vec{r} \)
- Recall Unit 5B (Iterative Methods), Steepest Descent for solving \( A x = b \) was
  - \( \vec{r}_k = b - A \vec{x}_k \), \( \alpha_k = \frac{\vec{r}_k \cdot \vec{r}_k}{\vec{r}_k \cdot A \vec{r}_k} \), \( \vec{x}_{k+1} = \vec{x}_k + \alpha_k \vec{r}_k \)
- The problem with steepest descent is that it repeatedly uses the same search directions, and thus takes a long time to converge, so we switched to (advocated) Conjugate Gradients

Hessian Approximation
- The cost function \( F(\vec{c}) \) is differentiated to obtain \( \vec{f}(\vec{c}) = \vec{F}(\vec{c}) = \vec{0} \), which is a nonlinear system specifying the critical points
- For line search, the typical linear solve for this nonlinear system is \( \vec{f}(\vec{c}^{k}) \vec{\Delta c} = \vec{\Delta F} \), which for an optimization problem is identical to \( \vec{H}(\vec{c}^{k}) \vec{\Delta c} = \vec{\Delta f} = \vec{0} - \vec{f}(\vec{c}^{k}) = -\vec{f}(\vec{c}^{k}) \)
- Steepest descent instead uses the search direction \( \vec{\Delta c} = -\nabla F(\vec{c}^{k}) = -\vec{f}(\vec{c}^{k}) \)
- I.e., steepest descent approximates \( \vec{H}(\vec{c}^k) = I \), which is the simplest possible approximation
Line Search

- Line search in the direction \( \tilde{c}(t) = \bar{c}^k + t \Delta c \)
- **Option 1** - use the search direction \( \Delta c \) to simultaneously look for *roots* of the vector valued nonlinear system \( \bar{f}(\tilde{c}(t)) = 0 \), which are the critical points of \( F(\tilde{c}) \), or to simultaneously *minimize* the \( \bar{f}(\tilde{c}(t)) \) aiming for roots

Each of the \( g(t) = \bar{f}_i(\tilde{c}(t)) \) has \( g'(t) = \bar{H}_i(\tilde{c}(t))\Delta c \) where \( \bar{H}_i(\tilde{c}(t)) \) is the i-th row of the Hessian of \( F \)

\[
\quad g''(t) = \Delta c^T \bar{OMG}_i(\tilde{c}(t)) \Delta c \quad \text{where} \quad \bar{OMG}_i \quad \text{is the i-th entry (which is a matrix) in a tensor} \quad \bar{OMG} \quad \text{of third derivatives and represents the Hessian of} \quad \bar{f}_i
\]

- **Option 2** - square the nonlinear system to obtain \( g(t) = \frac{1}{2} \bar{f}^T(\tilde{c}(t)) \bar{f}(\tilde{c}(t)) \) and look for *roots* of or *minimize* this 1D scalar function, again looking for the critical points of \( F(\tilde{c}) \)

\[
\quad g'(t) = \bar{f}^T(\tilde{c}(t)) \bar{H}(\tilde{c}(t)) \Delta c
\]

\[
\quad g''(t) = \Delta c^T \bar{H}(\tilde{c}(t)) \bar{H}(\tilde{c}(t)) \Delta c + \bar{f}^T(\tilde{c}(t)) \Delta c^T \bar{OMG}(\tilde{c}(t)) \Delta c + \bar{f}^T(\tilde{c}(t)) \Delta c^T \bar{OMG}(\tilde{c}(t)) \Delta c
\]

- **Option 3** – Minimize \( g(t) = F(\tilde{c}(t)) \) directly

\[
\quad g'(t) = \bar{f}^T(\tilde{c}(t)) \bar{H}(\tilde{c}(t)) \Delta c \quad \text{and} \quad g''(t) = \Delta c^T \bar{H}(\tilde{c}(t)) \Delta c
\]

Nonlinear Least Squares

- A *special cost function* is chosen of the form \( F(\bar{c}) = \frac{1}{2} f(\bar{c})^T f(\bar{c}) \) where \( f(\bar{c}) \) is a vector valued function

- Recall from Unit 13 that such a cost function emanated from trying to match a function with parameters \( \bar{c} \) to training data

\[
\quad F(\bar{c}) = \frac{1}{2} \sum_j (f_j(\bar{c}))^2 \quad \text{leads to the nonlinear system} \quad f(\bar{c}) = \sum_j f_j(\bar{c}) \frac{\partial f_j}{\partial \bar{c}}(\bar{c}) = f_j(\bar{c}) f(\bar{c}) = 0,
\]

where \( J_f(\bar{c}) \) is the Jacobian of \( f(\bar{c}) \) as opposed to the Jacobian of \( F(\tilde{c}) \)

- Steepest descent search direction is \( \Delta \bar{c} = -\nabla F(\bar{c}^k) = -f_j(\bar{c}^k) f(\bar{c}^k) \)

If there are a lot of terms in the summation for \( \Delta \bar{c} = -\sum_j f_j(\bar{c}^k) \frac{\partial f_j}{\partial \bar{c}}(\bar{c}^k) \) and the \( \frac{\partial f_j}{\partial \bar{c}}(\bar{c}^k) \) are expensive to compute (e.g. when using backward propagation), then sometimes only a subset of the terms are used in the summation

- **Stochastic Gradient Descent** uses only one term at a time, i.e. \( \Delta \bar{c} = -f_j(\bar{c}^k) \frac{\partial f_j}{\partial \bar{c}}(\bar{c}^k) \) for a randomly or sequentially chosen \( j \)

- **Mini-Batch Gradient Descent** uses a finite number of the terms instead of just one

In particular, for training data \( (\tilde{x}_i, \tilde{y}_i) \) and associated nonlinear least squares cost function \( F(\bar{c}) = \sum_i f(\tilde{x}_i, \tilde{y}_i, \bar{c})^T f(\tilde{x}_i, \tilde{y}_i, \bar{c}) \), Stochastic and Mini-Batch options can be seen as using only a subset of the training data to compute the search direction

- Using all the data is often called **Batch Gradient Descent**, which is identical to standard gradient descent