Appendix A
Optimization methods

A.1 Convexity

Definition A.1. (**Convex set**) We say a set $\mathcal{S}$ is convex if for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{S}$, we have

$$\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2 \in \mathcal{S}, \forall \lambda \in [0,1] \tag{A.1}$$

Definition A.2. (**Convex function**) A function $f(\boldsymbol{x})$ is called convex if its **epigraph**(the set of points above the function) defines a convex set. Equivalently, a function $f(\boldsymbol{x})$ is called convex if it is defined on a convex set and if, for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{S}$, and any $\lambda \in [0,1]$, we have

$$f(\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2) \leq \lambda f(\boldsymbol{x}_1) + (1-\lambda) f(\boldsymbol{x}_2) \tag{A.2}$$

Definition A.3. A function $f(\boldsymbol{x})$ is said to be **strictly convex** if the inequality is strict

$$f(\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2) < \lambda f(\boldsymbol{x}_1) + (1-\lambda) f(\boldsymbol{x}_2) \tag{A.3}$$

Definition A.4. A function $f(\boldsymbol{x})$ is said to be (strictly) **concave** if $-f(\boldsymbol{x})$ is (strictly) convex.

Theorem A.1. *If $f(x)$ is twice differentiable on $[a,b]$ and $f''(x) \geq 0$ on $[a,b]$ then $f(x)$ is convex on $[a,b]$.*

Proposition A.1. $\log(x)$ *is strictly convex on* $(0,\infty)$.

Intuitively, a (strictly) convex function has a bowl shape, and hence has a unique global minimum x^* corresponding to the bottom of the bowl. Hence its second derivative must be positive everywhere, $\frac{\mathrm{d}^2}{\mathrm{d}x^2} f(x) > 0$. A twice-continuously differentiable, multivariate function f is convex iff its Hessian is positive definite for all $\boldsymbol{x}$. In the machine learning context, the function f often corresponds to the NLL.

Models where the NLL is convex are desirable, since this means we can always find the globally optimal MLE. We will see many examples of this later in the book. However, many models of interest will not have concave likelihoods. In such cases, we will discuss ways to derive locally optimal parameter estimates.

A.2 Gradient descent

A.2.1 Stochastic gradient descent

input : Training data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) | i = 1 : N\}$
output: A linear model: $y_i = \boldsymbol{\theta}^T \boldsymbol{x}$
$\boldsymbol{w} \leftarrow 0$; $b \leftarrow 0$; $k \leftarrow 0$;
while *no mistakes made within the for loop* **do**
  for $i \leftarrow 1$ **to** N **do**
    if $y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \leq 0$ **then**
      $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta y_i \boldsymbol{x}_i$;
      $b \leftarrow b + \eta y_i$;
      $k \leftarrow k + 1$;
    end
  end
end

Algorithm 5: Stochastic gradient descent

A.2.2 Batch gradient descent

A.2.3 Line search

The **line search**[1] approach first finds a descent direction along which the objective function f will be reduced and then computes a step size that determines how far $\boldsymbol{x}$ should move along that direction. The descent direction can be computed by various methods, such as gradient descent(Section A.2), Newton's method(Section A.4) and Quasi-Newton method(Section A.5). The step size can be determined either exactly or inexactly.

[1] http://en.wikipedia.org/wiki/Line_search

A.2.4 Momentum term

A.3 Lagrange duality

A.3.1 Primal form

Consider the following, which we'll call the **primal** optimization problem:

$$xyz \tag{A.4}$$

A.3.2 Dual form

A.4 Newton's method

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}_k) + \boldsymbol{g}_k^T(\boldsymbol{x}-\boldsymbol{x}_k) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}_k)^T \boldsymbol{H}_k(\boldsymbol{x}-\boldsymbol{x}_k)$$

where $\boldsymbol{g}_k \triangleq \boldsymbol{g}(\boldsymbol{x}_k) = f'(\boldsymbol{x}_k), \boldsymbol{H}_k \triangleq \boldsymbol{H}(\boldsymbol{x}_k)$,

$$\boldsymbol{H}(\boldsymbol{x}) \triangleq \left[\frac{\partial^2 f}{\partial x_i \partial x_j}\right]_{D\times D} \quad \text{(Hessian matrix)}$$

$$f'(\boldsymbol{x}) = \boldsymbol{g}_k + \boldsymbol{H}_k(\boldsymbol{x}-\boldsymbol{x}_k) = 0 \Rightarrow \tag{A.5}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{H}_k^{-1}\boldsymbol{g}_k \tag{A.6}$$

Initialize $\boldsymbol{x}_0$
while *(!convergency)* **do**
  Evaluate $\boldsymbol{g}_k = \nabla f(\boldsymbol{x}_k)$
  Evaluate $\boldsymbol{H}_k = \nabla^2 f(\boldsymbol{x}_k)$
  $\boldsymbol{d}_k = -\boldsymbol{H}_k^{-1}\boldsymbol{g}_k$
  Use line search to find step size η_k along $\boldsymbol{d}_k$
  $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \eta_k \boldsymbol{d}_k$
end

Algorithm 6: Newtons method for minimizing a strictly convex function

A.5 Quasi-Newton method

From Equation A.5 we can infer out the **quasi-Newton condition** as follows:

$$\begin{aligned} f'(\boldsymbol{x}) - \boldsymbol{g}_k &= \boldsymbol{H}_k(\boldsymbol{x}-\boldsymbol{x}_k) \\ \boldsymbol{g}_{k-1} - \boldsymbol{g}_k &= \boldsymbol{H}_k(\boldsymbol{x}_{k-1}-\boldsymbol{x}_k) \Rightarrow \\ \boldsymbol{g}_k - \boldsymbol{g}_{k-1} &= \boldsymbol{H}_k(\boldsymbol{x}_k-\boldsymbol{x}_{k-1}) \\ \boldsymbol{g}_{k+1} - \boldsymbol{g}_k &= \boldsymbol{H}_{k+1}(\boldsymbol{x}_{k+1}-\boldsymbol{x}_k) \quad \text{(quasi-Newton condition)} \end{aligned} \tag{A.7}$$

The idea is to replace $\boldsymbol{H}_k^{-1}$ with a approximation $\boldsymbol{B}_k$, which satisfies the following properties:

1. $\boldsymbol{B}_k$ must be symmetric
2. $\boldsymbol{B}_k$ must satisfies the quasi-Newton condition, i.e., $\boldsymbol{g}_{k+1} - \boldsymbol{g}_k = \boldsymbol{B}_{k+1}(\boldsymbol{x}_{k+1}-\boldsymbol{x}_k)$.
   Let $\boldsymbol{y}_k = \boldsymbol{g}_{k+1} - \boldsymbol{g}_k$, $\boldsymbol{\delta}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$, then
   $$\boldsymbol{B}_{k+1}\boldsymbol{y}_k = \boldsymbol{\delta}_k \tag{A.8}$$
3. Subject to the above, $\boldsymbol{B}_k$ should be as close as possible to $\boldsymbol{B}_{k-1}$.

Note that we did not require that $\boldsymbol{B}_k$ be positive definite. That is because we can show that it must be positive definite if $\boldsymbol{B}_{k-1}$ is. Therefore, as long as the initial Hessian approximation $\boldsymbol{B}_0$ is positive definite, all $\boldsymbol{B}_k$ are, by induction.

A.5.1 DFP

Updating rule:

$$\boldsymbol{B}_{k+1} = \boldsymbol{B}_k + \boldsymbol{P}_k + \boldsymbol{Q}_k \tag{A.9}$$

From Equation A.8 we can get

$$\boldsymbol{B}_{k+1}\boldsymbol{y}_k = \boldsymbol{B}_k\boldsymbol{y}_k + \boldsymbol{P}_k\boldsymbol{y}_k + \boldsymbol{Q}_k\boldsymbol{y}_k = \boldsymbol{\delta}_k$$

To make the equation above establish, just let

$$\begin{aligned} \boldsymbol{P}_k\boldsymbol{y}_k &= \boldsymbol{\delta}_k \\ \boldsymbol{Q}_k\boldsymbol{y}_k &= -\boldsymbol{B}_k\boldsymbol{y}_k \end{aligned}$$

In DFP algorithm, $\boldsymbol{P}_k$ and $\boldsymbol{Q}_k$ are

$$\boldsymbol{P}_k = \frac{\boldsymbol{\delta}_k\boldsymbol{\delta}_k^T}{\boldsymbol{\delta}_k^T\boldsymbol{y}_k} \tag{A.10}$$

$$\boldsymbol{Q}_k = -\frac{\boldsymbol{B}_k\boldsymbol{y}_k\boldsymbol{y}_k^T\boldsymbol{B}_k}{\boldsymbol{y}_k^T\boldsymbol{B}_k\boldsymbol{y}_k} \tag{A.11}$$

A.5.2 BFGS

Use $\boldsymbol{B}_k$ as a approximation to $\boldsymbol{H}_k$, then the quasi-Newton condition becomes

$$\boldsymbol{B}_{k+1}\boldsymbol{\delta}_k = \boldsymbol{y}_k \tag{A.12}$$

The updating rule is similar to DFP, but $\boldsymbol{P}_k$ and $\boldsymbol{Q}_k$ are different. Let

$$\boldsymbol{P}_k \boldsymbol{\delta}_k = \boldsymbol{y}_k$$
$$\boldsymbol{Q}_k \boldsymbol{\delta}_k = -\boldsymbol{B}_k \boldsymbol{\delta}_k$$

Then

$$\boldsymbol{P}_k = \frac{\boldsymbol{y}_k \boldsymbol{y}_k^T}{\boldsymbol{y}_k^T \boldsymbol{\delta}_k} \tag{A.13}$$

$$\boldsymbol{Q}_k = -\frac{\boldsymbol{B}_k \boldsymbol{\delta}_k \boldsymbol{\delta}_k^T \boldsymbol{B}_k}{\boldsymbol{\delta}_k^T \boldsymbol{B}_k \boldsymbol{\delta}_k} \tag{A.14}$$

A.5.3 Broyden

Broyden's algorithm is a linear combination of DFP and BFGS.