STA 141C - Big Data & High Performance Statistical Computing **Spring 2022**

Week 8-1: Gradient descent

Lecturer: Bo Y.-C. Ning *May 18, 2022*

Disclaimer: *My notes may contain errors, distribution outside this class is allowed only with the permission of the Instructor.*

Last time

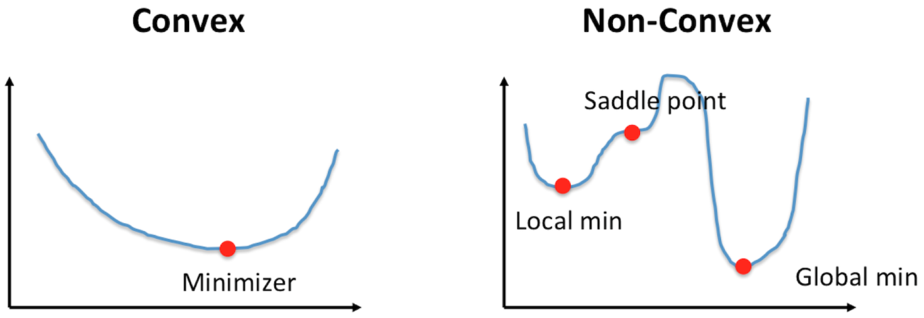
- Newton's algorithm

Today

- Gradient descent

1 Gradient descent

Our goal is to minimize a function $\min_\theta f(\theta)$, and let's assume the function f is twice differentiable. If f is a convex function, then $f'(\theta^\star) = 0$ implies that $\theta^\star$ is the global minimum. If f if s non-convex function, then $f'(\theta^\star) = 0$ implies that $\theta^\star$ can be the global minimum, local minimum, or a saddle point.



Last time, we introduced the Newton's method, in each iteration, we update

$$\theta^{(t+1)} = \theta^{(t)} - [f''(\theta^{(t)})]^{-1} f'(\theta^{(t)}).$$

One has to evaluate the inverse of the Hessian $f''(\theta^{(t)})$ each time, which can be expensive if the dimension of the Hessian matrix is large. Also, it can be reliable, e.g., using line search techniques.

The gradient descent replaces the Hessian matrix with sI, s is a scalar and I is the identity matrix. Then we switch to update

$$\theta^{(t+1)} = \theta^{(t)} - \frac{1}{s} f'(\theta^{(t)}).$$

In some textbooks, they let $\alpha = 1/s$ and call α the learning rate or step size.

Now we summarize the gradient descent algorithm:

```
Algorithm 1: Gradient descent
Input: Initial guess of $\theta^{(0)}$, step size $\alpha$, total iteration $T$, and $\epsilon$
For $t = 1, \dots, T$
Repeat:
        $\theta^{(t)} = \theta^{(t-1)} - \alpha f'(\theta^{(t-1)})$
Stop: If $\|\theta^{(t)} - \theta^{(t-1)}\| \leq \epsilon$ or $\|f'(\theta^{(t)})\| \leq \epsilon$
Output: $\hat{\theta} = \theta^{(t)}$
```

2 Examples

1. MLE for Poisson distribution

Suppose $x_1, \dots, x_n \sim \text{Poisson}(\lambda)$.

- Likelihood function: $f(x^n|\lambda) = \prod_{i=1}^n f(x_i|\lambda) = \prod_{i=1}^n \left[\lambda^{x_i} e^{-\lambda}/x_i!\,\right]$
- Log-likelihood function:

$$L(\lambda|x^n) = \log f(x^n|\lambda) = \sum_{i=1}^n x_i \log\lambda - n\lambda - \sum_{i=1}^n \log(x_i!\,)$$

- $dL(\lambda^{(t)}) = \frac{\sum_{i=1}^n x_i}{\lambda^{(t)}} - n$
- Update $\lambda^{(t+1)} = \lambda^{(t)} - \alpha\left(\frac{\sum_{i=1}^n x_i}{\lambda^{(t)}} - n\right) = \lambda^{(t)} - \frac{\alpha(\sum_{i=1}^n x_i - n\lambda^{(t)})}{\lambda^{(t)}}$.

2. Multivariate normal distribution

For a Gaussian model with a known variance Σ, $X_1, \dots, X_n \sim N(\theta, I_p)$,

- Likelihood function: $f(X^n|\theta) = \prod_{i=1}^n f(X_i|\theta) = \prod_{i=1}^n \left((\frac{1}{\sqrt{2\pi}})^p \exp(-(X_i-\theta)'(X_i-\theta)/2)\right)$
- $f(X^n|\theta) = \left(\frac{1}{\sqrt{2\pi}}\right)^{np} e^{-\frac{\sum_{i=1}^n (X_i-\theta)'(X_i-\theta)}{2}}$
- Log-likelihood function:

$$L(\theta \mid X^n) = \log f(X^n \mid \theta) = -\frac{np}{2}\log(2\pi) - \sum_{i=1}^n \frac{(X_i-\theta)'(X_i-\theta)}{2}.$$

- What is $dL(\theta^{(t)})$.
- $\theta = \theta^{(t)} - \alpha\sum_{i=1}^n (X_i - \theta^{(t)})$

3 Why and when does gradient descent work?

First, the gradient is the "steepest direction" to decrease the objective function locally. More formally, given a function f that is differentiable at x, the direction of the steepest direction is the vector $-f'(x)$.

Second, from the successive approximation view, at each iteration, the method forms an approximation function $f(\cdot)$. To see this,

$$\begin{aligned} f(\theta^{(t+1)}) = f(\theta^{(t)} + d) &\approx f(\theta^{(t)}) + f'(\theta^{(t)})'d + d'f''(\theta^{(t)})d/2 \\ &= f(\theta^{(t)}) + f'(\theta^{(t)})'d + d'd/(2\alpha) := g(d). \end{aligned}$$

Then update the solution $\theta^{(t+1)} \leftarrow \theta^{(t)} + d^\star$, such that $d^\star = \text{argmin}_d g(d)$. $g'(d) = 0 \Rightarrow d^\star = -\alpha f'(\theta^{(t)})$. At each step, $f(\cdot)$ decreases if α is sufficiently small.

In fact, one should choose the step size small enough such that $g(d) > f(\theta^{(t+1)} + d)$. In practice, the step size is unknown, one needs to tune step size when running gradient descent.

4 Handwriting recognition

We work on a model-based method for handwritten digit recognition. Following figure shows example bitmaps of handwritten digits from U.S. postal envelopes.

Each digit is represented by a 32×32 bitmap in which each element indicates one pixel with a value of white or black. Each 32×32 bitmap is divided into blocks of 4×4, and the number of white pixels are counted in each block. Therefore each handwritten digit is summarized by a vector $x = (x_1, \dots, x_{64})$ of length 64 where each element is a count between 0 and 16.

By a model-based method, we mean to impose a distribution on the count vector and carry out classification using probabilities. The goal is to predict handwritten digit. We separate the dataset into training data and test data. The training set contains 3823 handwritten digits and the test set contains 1797 digits.

A common distribution for count vectors is the multinomial distribution. However, it is not a good model for handwritten digits. Let's work on a more flexible model for count vectors. In the Dirichlet-multinomial model, we assume the multinomial probabilities $p = (p_1, \dots, p_d)$ follow a Dirichlet distribution with parameter vector $\alpha = (\alpha_1, \dots, \alpha_d)$, $\alpha_j > 0$. The density function is

$$f(p) = \frac{\Gamma(|\alpha|)}{\prod_{j=1}^d \Gamma(\alpha_j)} \prod_{j=1}^d p_j^{\alpha_j - 1},$$

where $|\alpha| = \sum_{j=1}^d \alpha_j$.

For a multivariate count vector $x = (x_1, \ldots, x_d)$ with batch size $|x| = \sum_{j=1}^d x_j$, the probability mass function for Dirichlet-multinomial distribution is

$$f(x \mid \alpha) = \binom{|x|}{x} \frac{\prod_{j=1}^d (\alpha_j)_{x_j}}{(|\alpha|)_{|x|}},$$

where $(\alpha_j)_{x_j} = \prod_{i=0}^{k-1}(a+i)$.

Given independent data points $x_1, \ldots, x_n$, the log-likelihood is given by

$$L(\alpha) = \sum_{i=1}^n \log \binom{|x_i|}{x_i} + \sum_{i=1}^n \sum_{j=1}^d [\log(\Gamma(\alpha_j + x_{ij}) - \log(\Gamma(\alpha_j))] - \sum_{i=1}^n [\log\Gamma(|\alpha| + |x_i|) - \log\Gamma(|\alpha|)] \,.$$

How do you calculate the MLE?

The score function is given by

$$\frac{\partial}{\partial \alpha_j} L(\alpha) = \sum_{i=1}^n [\Psi(x_{ij} + \alpha_j) - \Psi(\alpha_j)] - \sum_{i=1}^n [\Psi(|x_i| + |\alpha|) - \Psi(|\alpha|)] \,,$$

where $\Psi(x) = d(\log\Gamma(x)) = \Gamma'(x)/\Gamma(x)$.

The observed information matrix is

$$-d^2 L(\alpha) = D - c\mathbb{1}_d \mathbb{1}_d',$$

where D is a diagonal matrix,

$$D_{jj} = \sum_{i=1}^n [\Psi'(\alpha_j) - \Psi'(x_{ij} + \alpha_j)] \sum_{i=1}^n \sum_{k=0}^{x_{ij}-1} \frac{1}{(\alpha_j + k)^2}$$

and

$$c = \sum_{i=1}^n [\Psi'(|\alpha|) - \Psi'(|x_i| + |\alpha|)] = \sum_{i=1}^n \sum_{k=0}^{|x_i|-1} \frac{1}{(|\alpha| + k)^2}.$$

Then given an initial value for $\alpha^{(0)} = (\alpha_1^{(0)}, \ldots, \alpha_n^{(0)})$, the Newton's method keep updating

$$\alpha^{(t)} = \alpha^{(t-1)} + [-d^2 L(\alpha^{(t-1)})]^{-1} dL(\alpha^{(t-1)})$$

for $t = 1, \ldots, T$. We stop when $|L(\alpha^{(t)}) - L(\alpha^{(t-1)})| \leq \epsilon$, and then take $\hat{\alpha} = \alpha^{(t)}$.

Note that D is a diagonal matrix, we should use the Sherman-Morrison formula to write

$$[-d^2 L(\alpha)]^{-1} = D^{-1} + \frac{1}{1/c - \sum_j d_j^{-1}} D^{-1} \mathbb{1}\mathbb{1}' D^{-1}$$