OPTIMIZATION AND GRADIENT DESCENT BASICS

Shingchern D. You

Optimization

- Unconstrained optimization problem

  Ex: $\min f(x, y) = 3x - y^2$
- Constrained optimization problem

  Ex: $\min f(x, y)$ subject to $x^2 + y^2 = 1$
- It is easier to solve unconstrained optimization problem
- Some constrained optimization problems can be converted into unconstrained optimization problems

Optimization

- Finding optimal solution
  - Closed form
  - Iterative method (eg. gradient descent)
  - Randomized method (eg. genetic algorithm)

Unconstrained Optimization

- Closed form approach for one variable
- It is known that if a_0 is a local minimal/maximal, then $f'(a_0) = 0$
- We can find a_0 by solving $f'(x) = 0$ with additional verification steps
- The concept can be extended to multiple variables:

If (a_0, b_0) is a local minimal/maximal, then

$\frac{\partial}{\partial x} f(x, y)|_{x=a_0} = 0$ and $\frac{\partial}{\partial y} f(x, y)|_{y=b_0} = 0$

Unconstrained Optimization

- A multivariable function $f(x_1, \ldots, x_n)$ can also be written as $f(\mathbf{x})$ where $\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$
- We use $\nabla = \begin{bmatrix} \frac{\partial}{\partial x_1} \\ \vdots \\ \frac{\partial}{\partial x_n} \end{bmatrix}$ as the differential operator

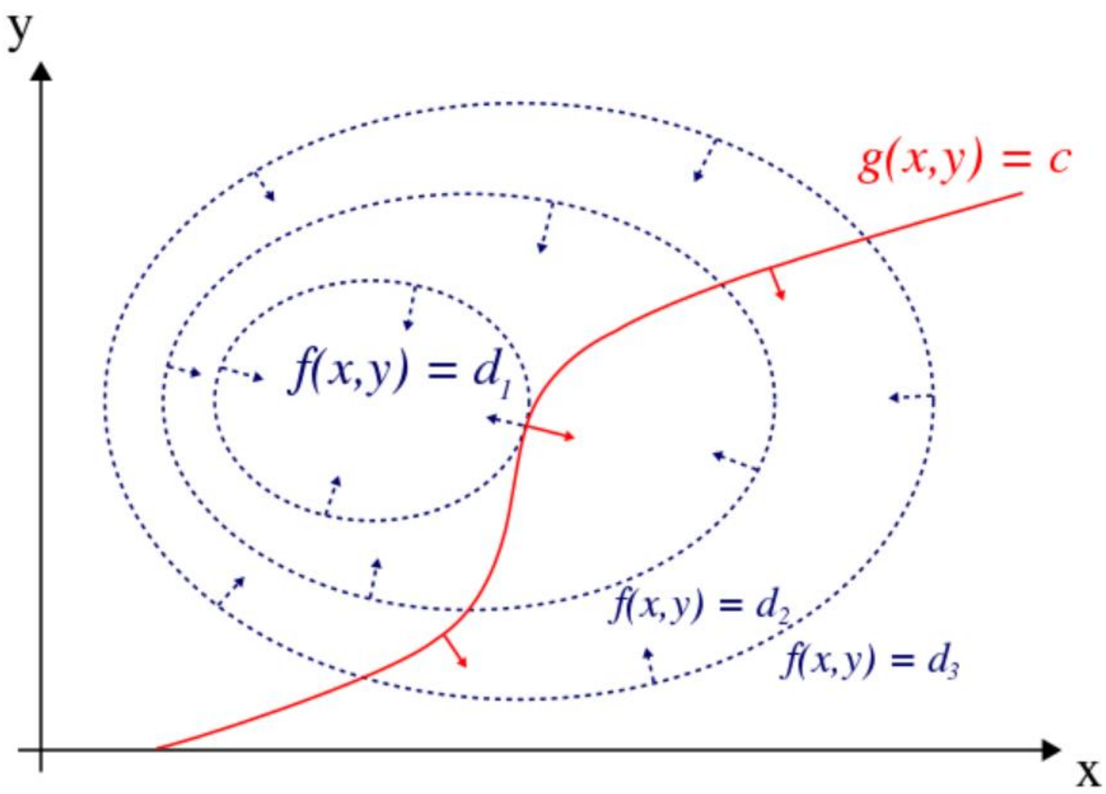
(also called Laplace operator or Laplacian)

Unconstrained Optimization

- If $\mathbf{x}_0$ is a vector of local minimal/maximal values, then $\nabla f(\mathbf{x}_0) = \mathbf{0}$
- The term ∇f is called gradient of f
- Again, we can find local extreme values by solving for $\nabla f(\mathbf{x}) = \mathbf{0}$ with additional verification steps

Constrained Optimization

- Want to maximize $f(x, y)$ subject to $g(x, y) = 0$
- Figure from https://en.wikipedia.org/wiki/Lagrange_multiplier



Constrained Optimization

- It is observed that the red line tangentially touches a blue contour is the maximum of $f(x, y)$ (note: the cotour of f is an uphill and $d_1 > d_2$)
- Thus, the gradient of f and g are parallel to each other
- Therefore, $\nabla f = \lambda \nabla g$ for some λ (λ is called Lagrange multiplier)
- Recall that maximal (minimal) point, $\nabla f = \mathbf{0}$. Thus, $\lambda \nabla g = \mathbf{0}$ too

Constrained Optimization

- The Lagrange multipliers method incorporate this fact in the auxiliary equation:

$$\mathcal{L}(x, y, \lambda) = f(x, y) + \lambda g(x, y)$$

  and solve the following

$$\frac{\partial}{\partial x}\mathcal{L} = 0, \frac{\partial}{\partial y}\mathcal{L} = 0, \frac{\partial}{\partial \lambda}\mathcal{L} = 0$$

- Note: $\frac{\partial}{\partial \lambda}\mathcal{L} = 0$ is the original constraint

Constrained Optimization

- Example from wiki
- Maximize $f(x, y) = x + y$ subject to $x^2 + y^2 = 1$
- $\mathcal{L}(x, y, \lambda) = x + y + \lambda(x^2 + y^2 - 1)$
- $\frac{\partial}{\partial x}\mathcal{L} = 1 + 2\lambda x = 0$
- $\frac{\partial}{\partial y}\mathcal{L} = 1 + 2\lambda y = 0$
- $\frac{\partial}{\partial \lambda}\mathcal{L} = x^2 + y^2 - 1 = 0$

Constrained Optimization

- Finally, we have the following stationary points

in the form of (x, y, λ)

$$\left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right) \text{ and } \left(-\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)$$

The first point is the max point

Constrained Optimization

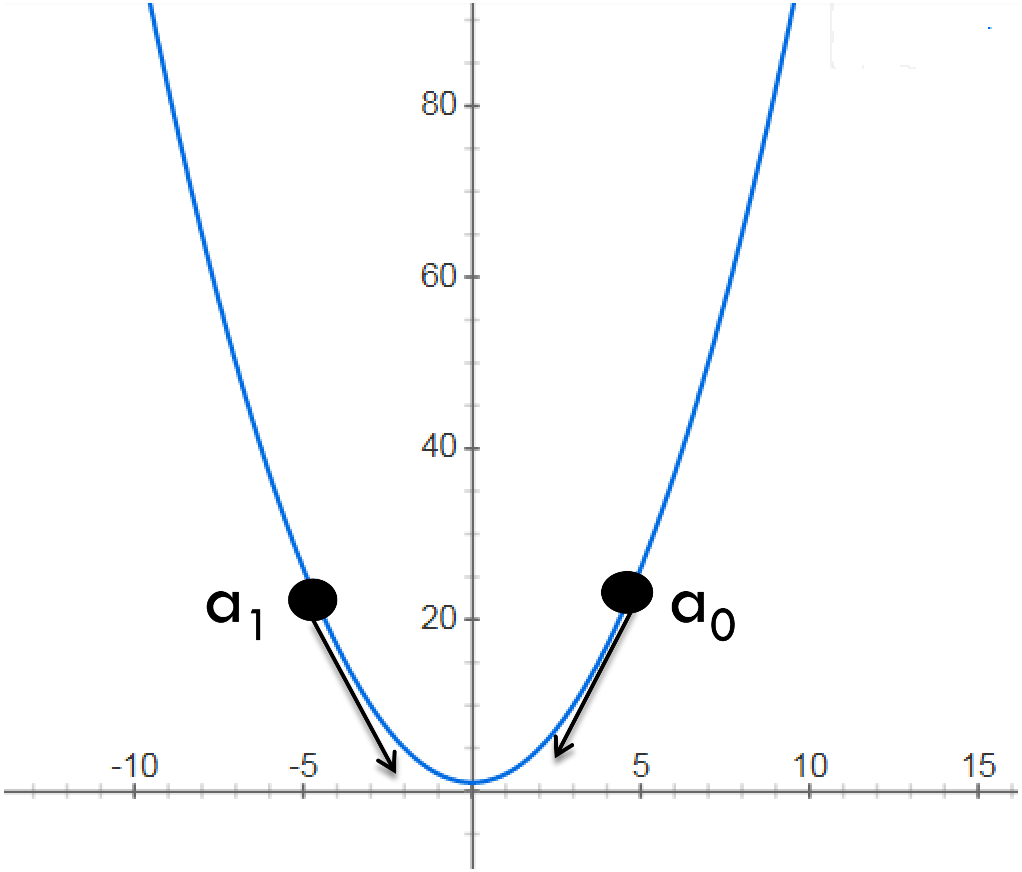
- Extension of Lagrange multipliers method to include inequality
- Optimize $f(x)$ subject to $g_i(x) \leq 0$ and $h_j(x) = 0$ $(i = 1..m, j = 1..n)$
- Karush–Kuhn–Tucker (KKT) conditions are first-order necessary conditions for a solution to above problem
- Use a method similar to Lagrange multipliers, $g_i(x) \leq 0$ constraints removed

Constrained Optimization

- KKT conditions can be extended to handle multivariable functions
- Usually solved by iterative methods
- Existing tools available for finding solutions for this type of problems
- Support vector machine (SVM) is based on KKT conditions

Gradient Descent Method

- Consider simple **unconstrained optimization** problem: $f(x) = x^2 + 1$



Gradient Descent Method

- Let Δ be a small number
- At point a_0, $f'(a_0) > 0$, if we want $f(a_0 + \Delta) < f(a_0)$, we know $\Delta < 0$
- At point a_1, $f'(a_1) < 0$, if we want $f(a_1 + \Delta) < f(a_1)$, we know $\Delta > 0$
- To find minimal value for function f, at $x = a_k$ we need to proceed with $\Delta = -\eta f'(a_k)$ where η is a small positive number

Gradient Descent Method

- Extend this idea, we have

$$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k - \eta \nabla f(\mathbf{x}_k)$$

where k is iteration index

- This approach is called gradient descent method (or gradient search)
- η is a small positive value
- How to find a "good" η for higher convergence is called "line search"

Gradient Descent Method

- Ref: http://theory.stanford.edu/~tim/s15/l/l15.pdf
- Another method to explain gradient descent method
- Consider a simple linear case ($\mathbf{x} \in R^p$)

$$f(\mathbf{x}) = \boldsymbol{w} \cdot \mathbf{x} + b$$

Where $\boldsymbol{w} \in R^p$ and $b \in R$ are constants and $\cdot$ denotes inner product, or $\boldsymbol{w} \cdot \mathbf{x} = \boldsymbol{w}^T\mathbf{x}$)

- Want to find vector $\boldsymbol{u} \in R^p$ with $\|\boldsymbol{u}\| = 1$ such that $f(\mathbf{x} + \boldsymbol{u})$ is minimal

Gradient Descent Method

- We know $f(\mathbf{x}+\boldsymbol{u}) = \boldsymbol{w}^{\boldsymbol{T}}(\mathbf{x}+\boldsymbol{u}) + b = f(\mathbf{x}) + \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{u}$
- Thus, $f(\mathbf{x}+\boldsymbol{u})$ is minimal only if $\boldsymbol{u} = -\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$
- In general, a multivariable function is not linear
- However, we can make it almost linear if considering small length of $\boldsymbol{u}$

Gradient Descent Method

- For small $\boldsymbol{u}$ (in length), we have Taylor's expansion for f about $\mathbf{x} = \mathbf{x}_0$ as
$$f(\mathbf{x}_0 + \boldsymbol{u}) \approx f(\mathbf{x}_0) + \boldsymbol{u}^T f(\mathbf{x}_0)'$$
- As the above equation is also a linear function, when compared with eq in previou slice, we have $\boldsymbol{w} = \nabla f(\mathbf{x}_0)$
- Because we want $\boldsymbol{u}$ to be small, minimal $f(\mathbf{x}_0 + \boldsymbol{u})$ occurs if $\boldsymbol{u}$ is in opposite direction of $\nabla f(\mathbf{x}_0)$,i.e., $\boldsymbol{u} = -\eta \nabla f(\mathbf{x}_0)$

Gradient Descent Method

- Algorithm to find minimal $f(\mathbf{x})$
  - Step 1: Find initial point $\mathbf{x}_0$. Let $k \leftarrow 0$
  - Step 2: While $\|\nabla f(\mathbf{x}_{\boldsymbol{k}})\| > \epsilon$ do

$$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k - \eta \nabla f(\mathbf{x}_k)$$
$$k \leftarrow k + 1$$

- In the algorithm, ϵ is a small positive number to determine the termination of the algorithm
- η is called step size and should be small to prevent problems

Gradient Descent Method

- There are lots of methods covering the selection of η. Simple algorithm uses a constant η throughout the iteration
- One way to determine this value is by line search which just means identifying by binary search the value of η that minimizing f over the line $\mathbf{x} - \eta \nabla f(\mathbf{x})$. After a few line searches, you should have a decent guess as to a good value of η

Estimate Gradient

- Sometimes it is not easy to compute gradient, we may use the following to estimate gradient

$$\frac{\partial}{\partial x_j} f(\mathbf{x}) \approx$$

$$\{f\left(x_1, \ldots, x_{j-1}, x_j + \eta, x_{j+1}, \ldots, x_p\right) - f(\mathbf{x})\}/\eta$$

- It is also useful to monitor the differences between true gradient and estimated gradient during iteration

Estimate Gradient

- Another (better) way to estimate gradient is through centered difference formula:

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h}$$ with small h

Gradient Descent for Regression

- Given a known dataset of $\boldsymbol{x}_{(\mathbf{1})}, \dots, \boldsymbol{x}_{(\boldsymbol{n})} \in R^p$ with corresponding labels $y_{(1)}, \dots, y_{(n)} \in R$, we define the error in the linear case as

$$\mathcal{E}_{(k)} = y_{(k)} - \boldsymbol{w}^{\boldsymbol{T}} \boldsymbol{x}_{(\boldsymbol{k})}, 1 \le k \le n$$

- If we want to extend to nonlinear function, we may use $\mathcal{E}_{(k)} = y_{(k)} - f\left(\boldsymbol{w}^{\boldsymbol{T}} \boldsymbol{x}_{(\boldsymbol{k})}\right)$

  where $f(\cdot)$ is a nonlinear function, such as sigmoid

Gradient Descent for Regression

- We can think a 2-class classification as a regression problem with $y_{(i)} \in \{0,1\}$, 0 for one class and 1 for another class
- What if we have multiple classes? A typical method is to use multiple 2-class classifiers
- If we have tree classes, we train three classifiers using three different settings for $y_{(i)}$. In class data have $y_{(i)} = 1$, all others are 0
- Alternatively, we may use softmax (covered later)

Gradient Descent for Regression

- We may define the cost function as

$$J(\boldsymbol{w}) = \frac{1}{n}\sum_{k=0}^{n} \varepsilon_{(k)}^2 = \frac{1}{n}\sum_{k=0}^{n} (y_{(k)} - \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_{(\boldsymbol{k})})^2$$

- Want to minimize the cost function by using gradient descent

Batch Gradient Descent

- It is easy to know

$$\nabla J(\boldsymbol{w}) = \frac{2}{n}\sum_{k=0}^{n}(y_{(k)} - \boldsymbol{w}^T\boldsymbol{x}_{(\boldsymbol{k})})\boldsymbol{x}_{(\boldsymbol{k})} = \frac{2}{n}\sum_{k=0}^{n}\varepsilon_{(k)}\boldsymbol{x}_{(\boldsymbol{k})}$$

- Note: average errors are used to determine gradient. Thus, it is called batch gradient descent
- Batch mode updates weights infrequently (**low efficiency)**
- If we have a nonlinear function $f(\cdot)$, we still can compute gradient by using chain rule

Stochastic Gradient Descent

- Another possible method to compute gradient for each instance $\boldsymbol{x}_{(\boldsymbol{k})}$ as

$$\nabla J(\boldsymbol{w}) = \varepsilon_{(k)} \boldsymbol{x}_{(\boldsymbol{k})}$$

- In stochastic gradient descent, weights updates for every instance
- In a typical situation, we nee to present the dataset to gradient descent many times
- Algorithm "learn" all data samples in training set one time is called one **epoch**

Stochastic Gradient Descent

- In stochastic gradient descent, step size must keep small. A large step size may **fail to converge**
- The adaptive signal processing version of stochastic gradient descent is called LMS (least mean squares) algorithm

Mini-batch Gradient Descent

- Recall
  - Batch gradient descent uses average error over entire dataset for one iteration (one gradient updating)
  - Stochastic gradient descent uses error on one instance for one iteration to update gradient
- We can do somewhere in between: Use average error of, say 128, instances for one weights update (called mini-batch gradient descent)

Sigmoid Function

- A linear function has limitations
- We can use nonlinear function to increase flexibility
- For example, a sigmoid function (also called logistic function) is

$$f(\mathbf{x}) = \frac{1}{1 + \exp(-\boldsymbol{w}^T\mathbf{x})}$$

- If we want, we can also add "bias" to equation, i.e., we use $(\boldsymbol{w}^T\mathbf{x} + w_0)$ in place of $\boldsymbol{w}^T\mathbf{x}$

Sigmoid Function

- A nice property of sigmoid function is simplicity on derivatives

$$\nabla f(\mathbf{x}) = \begin{bmatrix} f(x_1)(1 - f(x_1)) \\ \vdots \\ f(x_p)(1 - f(x_p)) \end{bmatrix}$$

where $f(x_j) = \frac{1}{1+\exp(-x_j)}$

- Note: $0 < f(x_j)(1 - f(x_j)) \leq \frac{1}{4}$

Other Nonlinear Function

- Another widely used nonlinear function is ReLU (Rectified Linear Unit)

$$f(x) = \max(0, x)$$

- ReLU does not suffer from "vanish of gradient" problem, a problem seen in sigmoid function

Regularization

- Recall we can avoid overfitting by using regularization
- We can do it in gradient descent with a modified cost function

$$J(\boldsymbol{w}) = \frac{1}{n}\sum_{k=0}^{n} \varepsilon_{(k)}^2 + \lambda g(\boldsymbol{w})$$

In the case of L-2 regularization, we have

$$g(\boldsymbol{w}) = \sum\nolimits_{j=1}^{p} w_j^2$$

Regularization

- We may also apply L-2 regularization to stochastic gradient descent (with slight modification)
- Exercise: How about mini-batch?
- Note: If "bias" term, i.e., w_0 in $(\boldsymbol{w}^T\mathbf{x} + w_0)$, is used, we will not penalize w_0
- Other than L-2 regularization, we may also try L-1 regularization (Lasso)
- Additional ref: https://arxiv.org/pdf/1609.04747.pdf

Momentum

- SGD has trouble navigating ravines, i.e. areas where the surface curves much more steeply in one dimension than in another, which are common around local optima
- Momentum is a method that helps accelerate SGD in the relevant direction and dampens oscillations

Momentum

- Weights updating with momentum

$$\Delta \boldsymbol{w}_{k+1} \leftarrow \eta \nabla f(\mathbf{x}_k) + \alpha \Delta \boldsymbol{w}_k$$
$$\boldsymbol{w}_{k+1} \leftarrow \boldsymbol{w}_k - \Delta \boldsymbol{w}_k$$

- Usually, $\alpha \approx 0.9$ is used
- In signal processing context, momentum is like a first-order IIR low-pass filter. Thus, $\Delta \boldsymbol{w}_k$ is smoothed
- More sophisticated methods such as Adam are available (suggested method for deep learning)

Further Reading

- As I mentioned previously, the textbook is way too dry (and difficult to appreciate for beginners)
- An excellent, easy to read reference material for (convolutional) neural networks is from Stanford university CS 231n at http://cs231n.stanford.edu/
- You may want to read the entire notes
- I will cover essential parts of that notes in our class