Linear Algebra and Optimization
MIT Fall 2022

Recitation 23

Tuesday December 6, 2022

1 Recap - Gradient Descent

Algorithm 1: Gradient Descent

Input: initial guess x_0, step size $\gamma > 0$;
while $\nabla f(x_k) \neq 0$ **do**
  $x_{k+1} = x_k - \gamma \nabla f(x_k)$
end
return x_k;

1.1 Convergence

1. Quadratic function: $\frac{1}{2}x^T Ax$ GD converges when the eigenvalues of $(I - \gamma A)$ have magnitude less than 1. Let λ_1 be the largest eigenvalue of A and λ_n be the smallest. Then when $0 < \gamma < \frac{2}{|\lambda_1|}$, GD converges.
   (a) Convergence rate $R = \|I - \gamma A\|$
   (b) $\gamma = \frac{2}{\lambda_1 + \lambda_n}$ gives the best (minimum) convergence rate.
   (c) $Q = \frac{|\lambda_1|}{|\lambda_n|}$ is the condition number of the matrix A. Smaller Q indicates more uniform convergence and hence faster convergence (lower rate). $R = \frac{Q-1}{Q+1}$.
2. Convex functions: More generally, we look at the eigenvalues $\lambda_1, ..., \lambda_n$ of the Hessian matrix.
   (a) Smooth function: $\lambda_i \leq L \ \forall i$. Choose step size $\gamma = \frac{1}{L}$. Then by the properties of f, $f(x_k) - f(x^*) \leq \frac{L}{2k}\|x_0 - x^*\|^2$
   (b) Smooth and Strongly Convex: $m \leq \lambda_i \leq L \ \forall i$. Condition number $Q = \frac{L}{m}$ Choose step size $\gamma = \frac{2}{m+L}$. Then $f(x_k) - f(x^*) \leq \frac{L}{2}\frac{Q-1}{Q+1}^{2k}\|x_0 - x^*\|^2$

1.2 Computing Gradient

1. Symbolic computation using rules of calculus
2. Numerical approximation using definition of derivatives
3. Automatic differentiation using composition of functions

1.3 Jacobian

The Jacobian matrix of a vector function $f : \mathbb{R}^n \to \mathbb{R}^m$ is the matrix of all its first-order partial derivatives. $\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_1} & \cdots & \frac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$

1.4 Stochastic GD

In Vanilla Gradient Descent (**Batch Gradient Descent**), we compute the gradient using all the data points in each iteration. This can be slow and redundant for large data sets. **Stochastic gradient descent** (SGD) in contrast performs gradient update for one randomly chosen training example at each iteration. Moreover, a method called **mini-batch gradient descent** combines both of them and computes the gradient based on a small batch of data points at each iteration.

1.5 Logistic Regression

Gradient descent is used for solving many machine learning problems, and one example is logistic regression. In the logistic regression model, we first apply a linear transformation on the input points x: $z = wx + b$. Then, we use the sigmoid function (aka logistic function) to classify the transformed points: $\sigma(z) = \frac{1}{1+e^{-z}}$.

1. Sigmoid
   The sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$ takes a real-valued number and maps it to the range [0,1], as if we are assigning a probability.

2. Decision Boundary
   We model $\sigma(z)$ as the probability that the corresponding label $y = 1$, and $1 - \sigma(z)$ as the probability that $y = 0$. Then we predict $\hat{y} = 1$ if $\sigma(z) > 0.5$, and $\hat{y} = 0$ otherwise.

3. Loss function
   Because of this relationship with probability, we choose a loss function called the cross entropy, or negative log likelihood:

   $$L(x, y; w) = -\sum y \log(\sigma(wx + b)) + (1 - y) \log(1 - \sigma(wx + b)).$$

   This loss function is conveniently convex, and we can find the global minimum using gradient descent.

2 Exercises

1. Find the gradient:
   (a) $f(x) = 4x_1x_2 + x_1^2 + 4x_2^2$
   (b) $g(x) = 2x_1x_2 + 2x_1^2 + 2x_2^2$
2. Find the Jacobian:
   (a) $F(x) = \begin{bmatrix} f(x) \\ g(x) \end{bmatrix}$
   (b) $\begin{matrix} x = r\cos\varphi; \\ y = r\sin\varphi. \end{matrix}$ Find the Jacobian with respect to r and φ
3. Are $f(x)$ and $g(x)$ smooth? Are they strongly convex?
4. Gradient Descent on $g(x)$:
   (a) What step size would you choose?
   (b) What is the condition number of the associated matrix?
   (c) What is the rate of convergence?
5. Logistic Regression example:
   Suppose we are doing binary sentiment classification on movie reviews, and we would like to know whether to assign the sentiment class + or - to each review. We use two features: the number of positive words and the number of negative words. We have three data points: ((3, 2), +), ((5, 1), -), ((2, 1), -). We want to develop a logistic regression model given these samples.
   (a) Suppose we have weights $w = (2, -1), b = -1$. What is our prediction on the first data point?
   (b) What is the gradient of the loss function?
   (c) Using stochastic gradient descent, what is the gradient after one iteration?
6. (Projected Gradient Descent). In this problem we explore gradient descent with convex constraints.
   (a) Let $S = \text{Span}(\{(1, 0)\})$. Write down a matrix A such that S is the nullspace of A.
   (b) Write down a quadratic optimization with linear constraints for finding the closest point in S to the point $(0, 1)$. That is come up with a function $f(x, y)$ such that
   $$\min_{(x,y)\in S} f(x, y)$$
   Write down the Hessian of this optimization.
   (c) If we move in the direction of the negative gradient $\nabla f(x, y)$ do we stay in S?
   (d) The projected gradient descent algorithm takes a step in the direction of the negative gradient $x^*_{t+1} = x_t - \gamma\nabla f(x_t)$ and then sets x_{t+1} to be the projection of x^*_{t+1} onto the subspace S. Write down the projected gradient descent update using A from part (a).

3 Solutions

1. (a) $\nabla f(x) = 2\begin{bmatrix}1 & 2\\ 2 & 4\end{bmatrix}x = Ax$

   (b) $\nabla g(x) = 2\begin{bmatrix}2 & 1\\ 1 & 2\end{bmatrix}x = Bx$

2. (a) $J(x) = \begin{bmatrix}2x^TA^T\\ 2x^TB^T\end{bmatrix}$

   (b)
   $$\mathbf{J}(r,\varphi) = \begin{bmatrix}\frac{\partial x}{\partial r} & \frac{\partial x}{\partial \varphi}\\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \varphi}\end{bmatrix} = \begin{bmatrix}\cos\varphi & -r\sin\varphi\\ \sin\varphi & r\cos\varphi\end{bmatrix}$$

3. A has eigenvalues 0,10. B has eigenvalues 2,6. $f(x)$ and $g(x)$ are both smooth. $g(x)$ is strongly convex,

4. (a) Since g is smooth and strongly convex, we can choose $\gamma = \frac{2}{2+6} = 0.25$.

   (b) The condition number of B is 3.

   (c) The rate of convergence is the largest singular value of $I - 2B$, which is 0.5.

5. (a) $z = wx + b = (6-2) - 1 = 3$. $\sigma(z) = 0.95 > 0.5 \Rightarrow \hat{y} = 1$

   (b) $\nabla L_w(x,y,w,b) = (\sigma(wx+b) - y)x$, $\nabla L_b(x,y,w,b) = \sigma(wx+b) - y$

   (c) Suppose we initialize $w = 0, b = 0$ and choose learning rate 0.1. Assume the first iteration we perform the update on the first data point. The gradient is $\nabla L_w(x,y,w,b) = \begin{bmatrix}-1.5\\ -1.0\end{bmatrix}$. So $w_1 = w_0 - 0.1L_w(x,y,w,b) = \begin{bmatrix}0.15\\ 0.1\end{bmatrix}$. And $b = 0 - (-1.5) = 1.5$

6. (a) $A = (0,1)$

   (b) $f(x,y) = x^2 + (y-1)^2$ with hessian $\begin{bmatrix}2 & 0\\ 0 & 2\end{bmatrix}$.

   (c) No, we leave S.

   (d)
   $$x^*_{t+1} = x_t - \gamma\nabla f(x_t)$$
   $$x_{t+1} = \begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}x^*_{t+1}$$