COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Cameron Musco

University of Massachusetts Amherst. Fall 2019.
Lecture 18

LOGISTICS

- Problem Set 3 on Spectral Methods due **this Friday at 8pm**.
- Can turn in without penalty until Sunday at 11:59pm.

SUMMARY

Last Class:

- Power method for computing the top singular vector of a matrix.
- High level discussion of Krylov methods, block versions for computing more singular vectors.

SUMMARY

Last Class:

- Power method for computing the top singular vector of a matrix.
- High level discussion of Krylov methods, block versions for computing more singular vectors.
- Power method is an iterative algorithm for solving the *non-convex* optimization problem:

$$\max_{\vec{v}:\|\vec{v}\|_2^2\leq 1} \vec{v}^T\mathbf{X}^T\mathbf{X}\vec{v}.$$

This Class (and until Thanksgiving):

- More general iterative algorithms for optimization, specifically gradient descent and its variants.
- What are they methods, when are they applied, and how do you analyze their performance?
- Small taste of what you can find in COMPSCI 590OP or 690OP.

DISCRETE VS. CONTINUOUS OPTIMIZATION

Discrete (Combinatorial) Optimization: (traditional CS algorithms)

- Graph Problems: min-cut, max flow, shortest path, matchings, maximum independent set, traveling salesman problem
- Problems with discrete constraints or outputs: bin-packing, scheduling, sequence alignment, submodular maximization
- Generally searching over a finite but exponentially large set of possible solutions. Many of these problems are NP-Hard.

DISCRETE VS. CONTINUOUS OPTIMIZATION

Discrete (Combinatorial) Optimization: (traditional CS algorithms)

- Graph Problems: min-cut, max flow, shortest path, matchings, maximum independent set, traveling salesman problem
- Problems with discrete constraints or outputs: bin-packing, scheduling, sequence alignment, submodular maximization
- Generally searching over a finite but exponentially large set of possible solutions. Many of these problems are NP-Hard.

DISCRETE VS. CONTINUOUS OPTIMIZATION

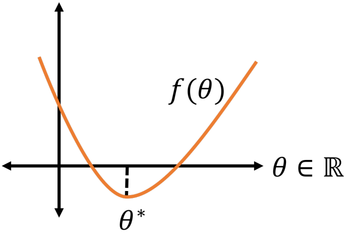
Discrete (Combinatorial) Optimization: (traditional CS algorithms)

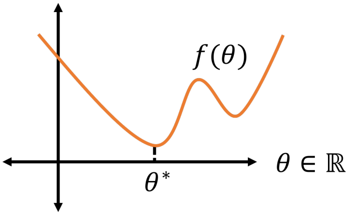
- Graph Problems: min-cut, max flow, shortest path, matchings, maximum independent set, traveling salesman problem
- Problems with discrete constraints or outputs: bin-packing, scheduling, sequence alignment, submodular maximization
- Generally searching over a finite but exponentially large set of possible solutions. Many of these problems are NP-Hard.

Continuous Optimization: (not covered in core CS curriculum. Touched on in ML/advanced algorithms, maybe.)

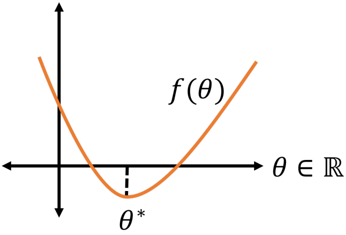
- Unconstrained convex and non-convex optimization.
- Linear programming, quadratic programming, semidefinite programming

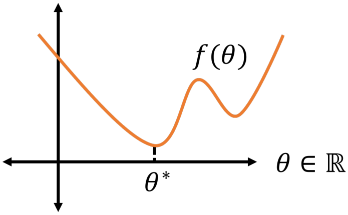
CONTINUOUS OPTIMIZATION EXAMPLES

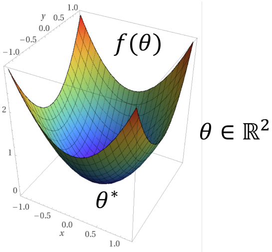


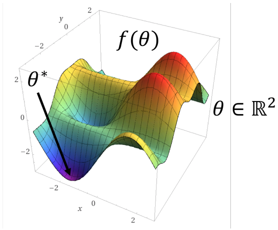


CONTINUOUS OPTIMIZATION EXAMPLES









MATHEMATICAL SETUP

Given some function $f : \mathbb{R}^d \to \mathbb{R}$, find $\vec{\theta}_\star$ with:

$$f(\vec{\theta}_\star) = \min_{\vec{\theta} \in R^d} f(\vec{\theta})$$

MATHEMATICAL SETUP

Given some function $f : \mathbb{R}^d \to \mathbb{R}$, find $\vec{\theta}_\star$ with:

$$f(\vec{\theta}_\star) = \min_{\vec{\theta} \in R^d} f(\vec{\theta}) + \epsilon$$

Typically up to some small approximation factor.

MATHEMATICAL SETUP

Given some function $f : \mathbb{R}^d \to \mathbb{R}$, find $\vec{\theta}_\star$ with:

$$f(\vec{\theta}_\star) = \min_{\vec{\theta} \in R^d} f(\vec{\theta}) + \epsilon$$

Typically up to some small approximation factor.

Often under some constraints:

- $\|\vec{\theta}\|_2 \leq 1, \quad \|\vec{\theta}\|_1 \leq 1.$
- $A\vec{\theta} \leq \vec{b}, \quad \vec{\theta}^T A\vec{\theta} \geq 0.$
- $\vec{1}^T\vec{\theta} = \sum_{i=1}^d \vec{\theta}(i) \leq c.$

WHY CONTINUOUS OPTIMIZATION?

WHY CONTINUOUS OPTIMIZATION?

Modern machine learning centers around continuous optimization.

WHY CONTINUOUS OPTIMIZATION?

Modern machine learning centers around continuous optimization.

Typical Set Up: (supervised machine learning)

- Have a model, which is a function mapping inputs to predictions (neural network, linear function, low-degree polynomial etc).
- The model is parameterized by a parameter vector (weights in a neural network, coefficients in a linear function or polynomial)
- Want to train this model on input data, by picking a parameter vector such that the model does a good job mapping inputs to predictions on your training data.

WHY CONTINUOUS OPTIMIZATION?

Modern machine learning centers around continuous optimization.

Typical Set Up: (supervised machine learning)

- Have a model, which is a function mapping inputs to predictions (neural network, linear function, low-degree polynomial etc).
- The model is parameterized by a parameter vector (weights in a neural network, coefficients in a linear function or polynomial)
- Want to train this model on input data, by picking a parameter vector such that the model does a good job mapping inputs to predictions on your training data.

This training step is typically formulated as a continuous optimization problem.

OPTIMIZATION IN ML

Example 1: Linear Regression

OPTIMIZATION IN ML

Example 1: Linear Regression

Model: $M_{\vec{\theta}} : \mathbb{R}^d \to \mathbb{R}$ with $M_{\vec{\theta}}(\vec{x}) \overset{\text{def}}{=} \langle \vec{\theta}, \vec{x} \rangle$.

OPTIMIZATION IN ML

Example 1: Linear Regression

Model: $M_{\vec{\theta}} : \mathbb{R}^d \to \mathbb{R}$ with $M_{\vec{\theta}}(\vec{x}) \overset{\text{def}}{=} \langle \vec{\theta}, \vec{x} \rangle = \vec{\theta}(1) \cdot \vec{x}(1) + \ldots + \vec{\theta}(d) \cdot \vec{x}(d)$.

OPTIMIZATION IN ML

Example 1: Linear Regression

Model: $M_{\vec{\theta}} : \mathbb{R}^d \to \mathbb{R}$ with $M_{\vec{\theta}}(\vec{x}) \stackrel{\text{def}}{=} \langle \vec{\theta}, \vec{x} \rangle = \vec{\theta}(1) \cdot \vec{x}(1) + \ldots + \vec{\theta}(d) \cdot \vec{x}(d)$.

Parameter Vector: $\vec{\theta} \in \mathbb{R}^d$ (the regression coefficients)

OPTIMIZATION IN ML

Example 1: Linear Regression

Model: $M_{\vec{\theta}} : \mathbb{R}^d \rightarrow \mathbb{R}$ with $M_{\vec{\theta}}(\vec{x}) \stackrel{\text{def}}{=} \langle \vec{\theta}, \vec{x} \rangle = \vec{\theta}(1) \cdot \vec{x}(1) + \ldots + \vec{\theta}(d) \cdot \vec{x}(d)$.

Parameter Vector: $\vec{\theta} \in \mathbb{R}^d$ (the regression coefficients)

Optimization Problem: Given data points (training points) $\vec{x}_1, \ldots, \vec{x}_n$ (the rows of data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$) and labels $y_1, \ldots, y_n \in \mathbb{R}$, find $\vec{\theta}_*$ minimizing the loss function:

$$L(\vec{\theta}, \mathbf{X}) = \sum_{i=1}^{n} \ell(M_{\vec{\theta}}(\vec{x}_i), y_i)$$

where ℓ is some measurement of how far $M_{\vec{\theta}}(\vec{x}_i)$ is from y_i.

OPTIMIZATION IN ML

Example 1: Linear Regression

Model: $M_{\vec{\theta}} : \mathbb{R}^d \rightarrow \mathbb{R}$ with $M_{\vec{\theta}}(\vec{x}) \stackrel{\text{def}}{=} \langle \vec{\theta}, \vec{x} \rangle = \vec{\theta}(1) \cdot \vec{x}(1) + \ldots + \vec{\theta}(d) \cdot \vec{x}(d)$.

Parameter Vector: $\vec{\theta} \in \mathbb{R}^d$ (the regression coefficients)

Optimization Problem: Given data points (training points) $\vec{x}_1, \ldots, \vec{x}_n$ (the rows of data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$) and labels $y_1, \ldots, y_n \in \mathbb{R}$, find $\vec{\theta}_*$ minimizing the loss function:

$$L(\vec{\theta}, \mathbf{X}) = \sum_{i=1}^{n} \ell(M_{\vec{\theta}}(\vec{x}_i), y_i)$$

where ℓ is some measurement of how far $M_{\vec{\theta}}(\vec{x}_i)$ is from y_i.

- $\ell(M_{\vec{\theta}}(\vec{x}_i), y_i) = \left(M_{\vec{\theta}}(\vec{x}_i) - y_i\right)^2$ (least squares regression)
- $y_i \in \{-1, 1\}$ and $\ell(M_{\vec{\theta}}(\vec{x}_i), y_i) = \ln\left(1 + \exp(-y_i M_{\vec{\theta}}(\vec{x}_i))\right)$ (logistic regression)

OPTIMIZATION IN ML

Example 1: Linear Regression

Model: $M_{\vec{\theta}} : \mathbb{R}^d \rightarrow \mathbb{R}$ with $M_{\vec{\theta}}(\vec{x}) \stackrel{\text{def}}{=} \langle \vec{\theta}, \vec{x} \rangle = \vec{\theta}(1) \cdot \vec{x}(1) + \ldots + \vec{\theta}(d) \cdot \vec{x}(d)$.

Parameter Vector: $\vec{\theta} \in \mathbb{R}^d$ (the regression coefficients)

Optimization Problem: Given data points (training points) $\vec{x}_1, \ldots, \vec{x}_n$ (the rows of data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$) and labels $y_1, \ldots, y_n \in \mathbb{R}$, find $\vec{\theta}_*$ minimizing the loss function:

$$L(\vec{\theta}, \mathbf{X}) = \sum_{i=1}^{n} \ell(M_{\vec{\theta}}(\vec{x}_i), y_i) + R(\vec{\theta})$$

where ℓ is some measurement of how far $M_{\vec{\theta}}(\vec{x}_i)$ is from y_i.

- $\ell(M_{\vec{\theta}}(\vec{x}_i), y_i) = \left(M_{\vec{\theta}}(\vec{x}_i) - y_i\right)^2$ (least squares regression)
- $y_i \in \{-1, 1\}$ and $\ell(M_{\vec{\theta}}(\vec{x}_i), y_i) = \ln\left(1 + \exp(-y_i M_{\vec{\theta}}(\vec{x}_i))\right)$ (logistic regression)

OPTIMIZATION IN ML

Example 1: Linear Regression

Model: $M_{\vec{\theta}} : \mathbb{R}^d \to \mathbb{R}$ with $M_{\vec{\theta}}(\vec{x}) \overset{\text{def}}{=} \langle \vec{\theta}, \vec{x} \rangle = \vec{\theta}(1) \cdot \vec{x}(1) + \ldots + \vec{\theta}(d) \cdot \vec{x}(d)$.

Parameter Vector: $\vec{\theta} \in \mathbb{R}^d$ (the regression coefficients)

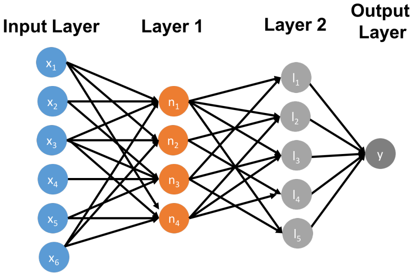
Optimization Problem: Given data points (training points) $\vec{x}_1, \ldots, \vec{x}_n$ (the rows of data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$) and labels $y_1, \ldots, y_n \in \mathbb{R}$, find $\vec{\theta}_*$ minimizing the loss function:

$$L(\vec{\theta}, \mathbf{X}) = \sum_{i=1}^{n} \ell(M_{\vec{\theta}}(\vec{x}_i), y_i) + \lambda \|\vec{\theta}\|_2^2$$

where ℓ is some measurement of how far $M_{\vec{\theta}}(\vec{x}_i)$ is from y_i.

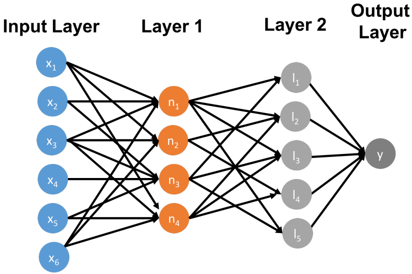
- $\ell(M_{\vec{\theta}}(\vec{x}_i), y_i) = \left(M_{\vec{\theta}}(\vec{x}_i) - y_i\right)^2$ (least squares regression)
- $y_i \in \{-1, 1\}$ and $\ell(M_{\vec{\theta}}(\vec{x}_i), y_i) = \ln\left(1 + \exp(-y_i M_{\vec{\theta}}(\vec{x}_i))\right)$ (logistic regression)

Example 2: Neural Networks



OPTIMIZATION IN ML

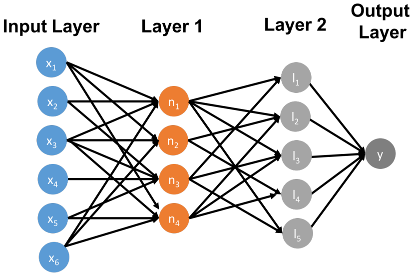
Example 2: Neural Networks



Model: $M_{\vec{\theta}} : \mathbb{R}^d \to \mathbb{R}$.

Parameter Vector: $\vec{\theta} \in \mathbb{R}^{(\#\ edges)}$ (the weights on every edge)

Example 2: Neural Networks

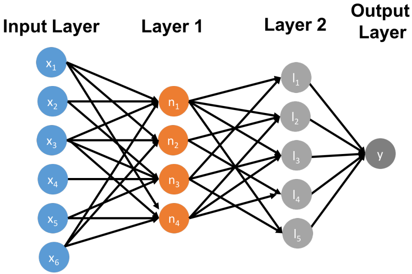


Model: $M_{\vec{\theta}} : \mathbb{R}^d \to \mathbb{R}$. $M_{\vec{\theta}}(\vec{x}) = \langle \vec{w}_{out}, \sigma(\mathbf{W}_2 \sigma(\mathbf{W}_1 \vec{x})) \rangle$.

Parameter Vector: $\vec{\theta} \in \mathbb{R}^{(\#\ edges)}$ (the weights on every edge)

OPTIMIZATION IN ML

Example 2: Neural Networks



Model: $M_{\vec{\theta}} : \mathbb{R}^d \to \mathbb{R}$. $M_{\vec{\theta}}(\vec{x}) = \langle \vec{w}_{out}, \sigma(\mathbf{W}_2 \sigma(\mathbf{W}_1 \vec{x})) \rangle$.

Parameter Vector: $\vec{\theta} \in \mathbb{R}^{(\#\ edges)}$ (the weights on every edge)

Optimization Problem: Given data points $\vec{x}_1, \ldots, \vec{x}_n$ and labels $y_1, \ldots, y_n \in \mathbb{R}$, find $\vec{\theta}_*$ minimizing the loss function:

$$L(\vec{\theta}, \mathbf{X}) = \sum_{i=1}^{n} \ell(M_{\vec{\theta}}(\vec{x}_i), y_i)$$

OPTIMIZATION IN ML

$$L(\vec{\theta}, \mathsf{X}) = \sum_{i=1}^{n} \ell(M_{\vec{\theta}}(\vec{x}_i), y_i)$$

- Supervised means we have labels $y_1, \ldots, y_n$ for the training points.

OPTIMIZATION IN ML

$$L(\vec{\theta}, \mathsf{X}) = \sum_{i=1}^{n} \ell(M_{\vec{\theta}}(\vec{x}_i), y_i)$$

- Supervised means we have labels $y_1, \ldots, y_n$ for the training points.
- Solving the final optimization problem has many different names: likelihood maximization, empirical risk minimization, minimizing training loss, etc.

OPTIMIZATION IN ML

$$L(\vec{\theta}, \mathbf{X}) = \sum_{i=1}^{n} \ell(M_{\vec{\theta}}(\vec{x}_i), y_i)$$

- Supervised means we have labels $y_1, \ldots, y_n$ for the training points.
- Solving the final optimization problem has many different names: likelihood maximization, empirical risk minimization, minimizing training loss, etc.
- Continuous optimization is also very common in unsupervised learning.

OPTIMIZATION IN ML

$$L(\vec{\theta}, \mathbf{X}) = \sum_{i=1}^{n} \ell(M_{\vec{\theta}}(\vec{x}_i), y_i)$$

- Supervised means we have labels $y_1, \ldots, y_n$ for the training points.
- Solving the final optimization problem has many different names: likelihood maximization, empirical risk minimization, minimizing training loss, etc.
- Continuous optimization is also very common in unsupervised learning. (PCA, spectral clustering, etc.)

OPTIMIZATION IN ML

$$L(\vec{\theta}, \mathrm{X}) = \sum_{i=1}^{n} \ell(M_{\vec{\theta}}(\vec{x}_i), y_i)$$

- Supervised means we have labels $y_1, \ldots, y_n$ for the training points.
- Solving the final optimization problem has many different names: likelihood maximization, empirical risk minimization, minimizing training loss, etc.
- Continuous optimization is also very common in unsupervised learning. (PCA, spectral clustering, etc.)
- Generalization tries to explain why minimizing the loss $L(\vec{\theta}, \mathrm{X})$ on the *training points* minimizes the loss on future *test points*. I.e., makes us have good predictions on future inputs.

OPTIMIZATION ALGORITHMS

Choice of optimization algorithm for minimizing $f(\vec{\theta})$ will depend on many things:

- The form of f (in ML, depends on the model & loss function).
- Any constraints on $\vec{\theta}$ (e.g., $\|\vec{\theta}\| < c$).
- Other constraints, such as memory constraints.

OPTIMIZATION ALGORITHMS

Choice of optimization algorithm for minimizing $f(\vec{\theta})$ will depend on many things:

- The form of f (in ML, depends on the model & loss function).
- Any constraints on $\vec{\theta}$ (e.g., $\|\vec{\theta}\| < c$).
- Other constraints, such as memory constraints.

$$L(\vec{\theta}, \mathbf{X}) = \sum_{i=1}^{n} \ell(M_{\vec{\theta}}(\vec{x}_i), y_i)$$

What are some popular optimization algorithms?

GRADIENT DESCENT

This class: Gradient descent (and some important variants)

GRADIENT DESCENT

This class: Gradient descent (and some important variants)

- An extremely simple greedy iterative method, that can be applied to almost any continuous function we care about optimizing.

GRADIENT DESCENT

This class: Gradient descent (and some important variants)

- An extremely simple greedy iterative method, that can be applied to almost any continuous function we care about optimizing.
- Often not the ‘best’ choice for any given function, but it is the approach of choice in ML since it is simple, general, and often works very well.

GRADIENT DESCENT

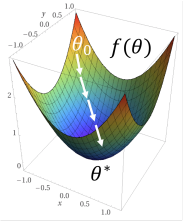
This class: Gradient descent (and some important variants)

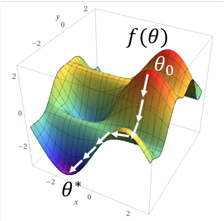
- An extremely simple greedy iterative method, that can be applied to almost any continuous function we care about optimizing.
- Often not the ‘best’ choice for any given function, but it is the approach of choice in ML since it is simple, general, and often works very well.
- At each step, tries to move towards the lowest nearby point in the function that is can – in the direction of the gradient.

GRADIENT DESCENT

This class: Gradient descent (and some important variants)

- An extremely simple greedy iterative method, that can be applied to almost any continuous function we care about optimizing.
- Often not the ‘best’ choice for any given function, but it is the approach of choice in ML since it is simple, general, and often works very well.
- At each step, tries to move towards the lowest nearby point in the function that is can – in the direction of the gradient.





MULTIVARIATE CALCULUS REVIEW

Let $\vec{e}_i \in \mathbb{R}^d$ denote the i^{th} standard basis vector, $\vec{e}_i = \underbrace{[0, 0, 1, 0, 0, \ldots, 0]}_{\text{1 at position } i}$.

MULTIVARIATE CALCULUS REVIEW

Let $\vec{e}_i \in \mathbb{R}^d$ denote the i^{th} standard basis vector, $\vec{e}_i = \underbrace{[0, 0, 1, 0, 0, \ldots, 0]}_{\text{1 at position } i}$.

Partial Derivative:

$$\frac{\partial f}{\partial \vec{\theta}(i)} = \lim_{\epsilon \to 0} \frac{f(\vec{\theta} + \epsilon \cdot \vec{e}_i) - f(\vec{\theta})}{\epsilon}.$$

MULTIVARIATE CALCULUS REVIEW

Let $\vec{e}_i \in \mathbb{R}^d$ denote the i^{th} standard basis vector, $\vec{e}_i = \underbrace{[0, 0, 1, 0, 0, \ldots, 0]}_{\text{1 at position } i}$.

Partial Derivative:

$$\frac{\partial f}{\partial \vec{\theta}(i)} = \lim_{\epsilon \to 0} \frac{f(\vec{\theta} + \epsilon \cdot \vec{e}_i) - f(\vec{\theta})}{\epsilon}.$$

Directional Derivative:

$$D_{\vec{v}} f(\vec{\theta}) = \lim_{\epsilon \to 0} \frac{f(\vec{\theta} + \epsilon \vec{v}) - f(\vec{\theta})}{\epsilon}.$$

MULTIVARIATE CALCULUS REVIEW

Gradient: Just a 'list' of the partial derivatives.

$$\vec{\nabla} f(\vec{\theta}) = \begin{bmatrix} \frac{\partial f}{\partial \vec{\theta}(1)} \\ \frac{\partial f}{\partial \vec{\theta}(2)} \\ \vdots \\ \frac{\partial f}{\partial \vec{\theta}(d)} \end{bmatrix}$$

MULTIVARIATE CALCULUS REVIEW

Gradient: Just a 'list' of the partial derivatives.

$$\vec{\nabla} f(\vec{\theta}) = \begin{bmatrix} \frac{\partial f}{\partial \vec{\theta}(1)} \\ \frac{\partial f}{\partial \vec{\theta}(2)} \\ \vdots \\ \frac{\partial f}{\partial \vec{\theta}(d)} \end{bmatrix}$$

Directional Derivative in Terms of the Gradient:

$$D_{\vec{v}}\, f(\vec{\theta}) = \lim_{\epsilon \to 0} \frac{f(\vec{\theta} + \epsilon \vec{v}) - f(\vec{\theta})}{\epsilon}$$

MULTIVARIATE CALCULUS REVIEW

Gradient: Just a 'list' of the partial derivatives.

$$\vec{\nabla} f(\vec{\theta}) = \begin{bmatrix} \frac{\partial f}{\partial \vec{\theta}(1)} \\ \frac{\partial f}{\partial \vec{\theta}(2)} \\ \vdots \\ \frac{\partial f}{\partial \vec{\theta}(d)} \end{bmatrix}$$

Directional Derivative in Terms of the Gradient:

$$D_{\vec{v}}\, f(\vec{\theta}) = \lim_{\epsilon \to 0} \frac{f(\vec{\theta} + \epsilon(\vec{e}_1 \cdot \vec{v}(1) + \vec{e}_2 \cdot \vec{v}(2) + \ldots + \vec{e}_d \cdot \vec{v}(d)) - f(\vec{\theta})}{\epsilon}$$

MULTIVARIATE CALCULUS REVIEW

Gradient: Just a 'list' of the partial derivatives.

$$\vec{\nabla} f(\vec{\theta}) = \begin{bmatrix} \frac{\partial f}{\partial \vec{\theta}(1)} \\ \frac{\partial f}{\partial \vec{\theta}(2)} \\ \vdots \\ \frac{\partial f}{\partial \vec{\theta}(d)} \end{bmatrix}$$

Directional Derivative in Terms of the Gradient:

$$D_{\vec{v}}\, f(\vec{\theta}) = \lim_{\epsilon \to 0} \frac{f(\vec{\theta} + \epsilon(\vec{e}_1 \cdot \vec{v}(1) + \vec{e}_2 \cdot \vec{v}(2) + \ldots + \vec{e}_d \cdot \vec{v}(d)) - f(\vec{\theta})}{\epsilon}$$

MULTIVARIATE CALCULUS REVIEW

Gradient: Just a 'list' of the partial derivatives.

$$\vec{\nabla} f(\vec{\theta}) = \begin{bmatrix} \frac{\partial f}{\partial \vec{\theta}(1)} \\ \frac{\partial f}{\partial \vec{\theta}(2)} \\ \vdots \\ \frac{\partial f}{\partial \vec{\theta}(d)} \end{bmatrix}$$

Directional Derivative in Terms of the Gradient:

$$\begin{aligned} D_{\vec{v}}\, f(\vec{\theta}) &= \lim_{\epsilon \to 0} \frac{f(\vec{\theta} + \epsilon(\vec{e}_1 \cdot \vec{v}(1) + \vec{e}_2 \cdot \vec{v}(2) + \ldots + \vec{e}_d \cdot \vec{v}(d)) - f(\vec{\theta})}{\epsilon} \\ &\approx \vec{v}(1) \cdot \frac{\partial f}{\partial \vec{\theta}(1)} \end{aligned}$$

MULTIVARIATE CALCULUS REVIEW

Gradient: Just a 'list' of the partial derivatives.

$$\vec{\nabla} f(\vec{\theta}) = \begin{bmatrix} \frac{\partial f}{\partial \vec{\theta}(1)} \\ \frac{\partial f}{\partial \vec{\theta}(2)} \\ \vdots \\ \frac{\partial f}{\partial \vec{\theta}(d)} \end{bmatrix}$$

Directional Derivative in Terms of the Gradient:

$$\begin{aligned} D_{\vec{v}} f(\vec{\theta}) &= \lim_{\epsilon \to 0} \frac{f(\vec{\theta} + \epsilon(\vec{e}_1 \cdot \vec{v}(1) + \vec{e}_2 \cdot \vec{v}(2) + \ldots + \vec{e}_d \cdot \vec{v}(d)) - f(\vec{\theta})}{\epsilon} \\ &\approx \vec{v}(1) \cdot \frac{\partial f}{\partial \vec{\theta}(1)} + \vec{v}(2) \cdot \frac{\partial f}{\partial \vec{\theta}(2)} + \ldots + \vec{v}(d) \cdot \frac{\partial f}{\partial \vec{\theta}(d)} \end{aligned}$$

MULTIVARIATE CALCULUS REVIEW

Gradient: Just a 'list' of the partial derivatives.

$$\vec{\nabla} f(\vec{\theta}) = \begin{bmatrix} \frac{\partial f}{\partial \vec{\theta}(1)} \\ \frac{\partial f}{\partial \vec{\theta}(2)} \\ \vdots \\ \frac{\partial f}{\partial \vec{\theta}(d)} \end{bmatrix}$$

Directional Derivative in Terms of the Gradient:

$$\begin{aligned} D_{\vec{v}} f(\vec{\theta}) &= \lim_{\epsilon \to 0} \frac{f(\vec{\theta} + \epsilon(\vec{e}_1 \cdot \vec{v}(1) + \vec{e}_2 \cdot \vec{v}(2) + \ldots + \vec{e}_d \cdot \vec{v}(d)) - f(\vec{\theta})}{\epsilon} \\ &\approx \vec{v}(1) \cdot \frac{\partial f}{\partial \vec{\theta}(1)} + \vec{v}(2) \cdot \frac{\partial f}{\partial \vec{\theta}(2)} + \ldots + \vec{v}(d) \cdot \frac{\partial f}{\partial \vec{\theta}(d)} \\ &= \langle \vec{v}, \vec{\nabla} f(\vec{\theta}) \rangle. \end{aligned}$$

FUNCTION ACCESS

Often the functions we are trying to optimize are very complex (e.g., a neural network). We will assume access to:

Function Evaluation: Can compute $f(\vec{\theta})$ for any $\vec{\theta}$.

Gradient Evaluation: Can compute $\vec{\nabla} f(\vec{\theta})$ for any $\vec{\theta}$.

FUNCTION ACCESS

Often the functions we are trying to optimize are very complex (e.g., a neural network). We will assume access to:

Function Evaluation: Can compute $f(\vec{\theta})$ for any $\vec{\theta}$.

Gradient Evaluation: Can compute $\vec{\nabla} f(\vec{\theta})$ for any $\vec{\theta}$.

In neural networks:

- Function evaluation is called a forward pass (propogate an input through the network).
- Gradient evaluation is called a backward pass (compute the gradient via chain rule, using backpropagation).

GRADIENT EXAMPLE

Running Example: Least squares regression.

Given input points $\vec{x}_1, \ldots \vec{x}_n$ (the rows of data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$) and labels $y_1, \ldots, y_n$ (the entries of $\vec{y} \in \mathbb{R}^n$) , find $\vec{\theta}_*$ minimizing:

$$L(\vec{\theta}, \mathbf{X}) = \sum_{i=1}^{n} \left(\vec{\theta}^T \vec{x}_i - y_i \right)^2$$

GRADIENT EXAMPLE

Running Example: Least squares regression.

Given input points $\vec{x}_1, \ldots \vec{x}_n$ (the rows of data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$) and labels $y_1, \ldots, y_n$ (the entries of $\vec{y} \in \mathbb{R}^n$) , find $\vec{\theta}_*$ minimizing:

$$L(\vec{\theta}, \mathbf{X}) = \sum_{i=1}^{n} \left(\vec{\theta}^T \vec{x}_i - y_i\right)^2 = \|\mathbf{X}\vec{\theta} - \vec{y}\|_2^2.$$

By Chain rule:

$$\frac{\partial L(\vec{\theta}, \mathbf{X})}{\partial \vec{\theta}(j)} = \sum_{i=1}^{n} 2 \cdot \left(\vec{\theta}^T \vec{x}_i - y_i\right) \cdot \frac{\partial \left(\vec{\theta}^T \vec{x}_i - y_i\right)}{\partial \vec{\theta}(j)}$$

GRADIENT EXAMPLE

Running Example: Least squares regression.

Given input points $\vec{x}_1, \ldots \vec{x}_n$ (the rows of data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$) and labels $y_1, \ldots, y_n$ (the entries of $\vec{y} \in \mathbb{R}^n$) , find $\vec{\theta}_*$ minimizing:

$$L(\vec{\theta}, \mathbf{X}) = \sum_{i=1}^{n} \left(\vec{\theta}^T \vec{x}_i - y_i\right)^2 = \|\mathbf{X}\vec{\theta} - \vec{y}\|_2^2.$$

By Chain rule:

$$\frac{\partial L(\vec{\theta}, \mathbf{X})}{\partial \vec{\theta}(j)} = \sum_{i=1}^{n} 2 \cdot \left(\vec{\theta}^T \vec{x}_i - y_i\right) \cdot \frac{\partial \left(\vec{\theta}^T \vec{x}_i - y_i\right)}{\partial \vec{\theta}(j)}$$

$$\frac{\partial \left(\vec{\theta}^T \vec{x}_i - y_i\right)}{\partial \vec{\theta}(j)} = \frac{\partial (\theta^T \vec{x}_i)}{\partial \vec{\theta}(j)}$$

GRADIENT EXAMPLE

Running Example: Least squares regression.

Given input points $\vec{x}_1, \ldots \vec{x}_n$ (the rows of data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$) and labels $y_1, \ldots, y_n$ (the entries of $\vec{y} \in \mathbb{R}^n$) , find $\vec{\theta}_*$ minimizing:

$$L(\vec{\theta}, \mathbf{X}) = \sum_{i=1}^{n} \left(\vec{\theta}^T \vec{x}_i - y_i \right)^2 = \|\mathbf{X}\vec{\theta} - \vec{y}\|_2^2.$$

By Chain rule:

$$\frac{\partial L(\vec{\theta}, \mathbf{X})}{\partial \vec{\theta}(j)} = \sum_{i=1}^{n} 2 \cdot \left(\vec{\theta}^T \vec{x}_i - y_i \right) \cdot \frac{\partial \left(\vec{\theta}^T \vec{x}_i - y_i \right)}{\partial \vec{\theta}(j)}$$

$$\frac{\partial \left(\vec{\theta}^T \vec{x}_i - y_i \right)}{\partial \vec{\theta}(j)} = \frac{\partial (\theta^T \vec{x}_i)}{\partial \vec{\theta}(j)} = \lim_{\epsilon \to 0} \frac{\theta^T \vec{x}_i - (\theta + \epsilon \vec{e}_j)^T \vec{x}_i}{\epsilon}$$

GRADIENT EXAMPLE

Running Example: Least squares regression.

Given input points $\vec{x}_1, \ldots \vec{x}_n$ (the rows of data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$) and labels $y_1, \ldots, y_n$ (the entries of $\vec{y} \in \mathbb{R}^n$) , find $\vec{\theta}_*$ minimizing:

$$L(\vec{\theta}, \mathbf{X}) = \sum_{i=1}^{n} \left(\vec{\theta}^T \vec{x}_i - y_i \right)^2 = \|\mathbf{X}\vec{\theta} - \vec{y}\|_2^2.$$

By Chain rule:

$$\frac{\partial L(\vec{\theta}, \mathbf{X})}{\partial \vec{\theta}(j)} = \sum_{i=1}^{n} 2 \cdot \left(\vec{\theta}^T \vec{x}_i - y_i \right) \cdot \frac{\partial \left(\vec{\theta}^T \vec{x}_i - y_i \right)}{\partial \vec{\theta}(j)}$$

$$\frac{\partial \left(\vec{\theta}^T \vec{x}_i - y_i \right)}{\partial \vec{\theta}(j)} = \frac{\partial (\theta^T \vec{x}_i)}{\partial \vec{\theta}(j)} = \lim_{\epsilon \to 0} \frac{\theta^T \vec{x}_i - (\theta + \epsilon \vec{e}_j)^T \vec{x}_i}{\epsilon} = \lim_{\epsilon \to 0} \frac{\epsilon \vec{e}_j^T \vec{x}_i}{\epsilon}$$

GRADIENT EXAMPLE

Running Example: Least squares regression.

Given input points $\vec{x}_1, \ldots \vec{x}_n$ (the rows of data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$) and labels $y_1, \ldots, y_n$ (the entries of $\vec{y} \in \mathbb{R}^n$) , find $\vec{\theta}_*$ minimizing:

$$L(\vec{\theta}, \mathbf{X}) = \sum_{i=1}^{n} \left(\vec{\theta}^T \vec{x}_i - y_i \right)^2 = \|\mathbf{X}\vec{\theta} - \vec{y}\|_2^2.$$

By Chain rule:

$$\frac{\partial L(\vec{\theta}, \mathbf{X})}{\partial \vec{\theta}(j)} = \sum_{i=1}^{n} 2 \cdot \left(\vec{\theta}^T \vec{x}_i - y_i \right) \cdot \frac{\partial \left(\vec{\theta}^T \vec{x}_i - y_i \right)}{\partial \vec{\theta}(j)}$$

$$\frac{\partial \left(\vec{\theta}^T \vec{x}_i - y_i \right)}{\partial \vec{\theta}(j)} = \frac{\partial (\theta^T \vec{x}_i)}{\partial \vec{\theta}(j)} = \lim_{\epsilon \to 0} \frac{\theta^T \vec{x}_i - (\theta + \epsilon \vec{e}_j)^T \vec{x}_i}{\epsilon} = \lim_{\epsilon \to 0} \frac{\epsilon \vec{e}_j^T \vec{x}_i}{\epsilon} = \vec{x}_i(j).$$

GRADIENT EXAMPLE

Running Example: Least squares regression.

Given input points $\vec{x}_1, \ldots \vec{x}_n$ (the rows of data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$) and labels $y_1, \ldots, y_n$ (the entries of $\vec{y} \in \mathbb{R}^n$) , find $\vec{\theta}_*$ minimizing:

$$L(\vec{\theta}, \mathbf{X}) = \sum_{i=1}^{n} \left(\vec{\theta}^T \vec{x}_i - y_i \right)^2 = \|\mathbf{X}\vec{\theta} - \vec{y}\|_2^2.$$

By Chain rule:

$$\frac{\partial L(\vec{\theta}, \mathbf{X})}{\partial \vec{\theta}(j)} = \sum_{i=1}^{n} 2 \cdot \left(\vec{\theta}^T \vec{x}_i - y_i \right) \cdot \frac{\partial \left(\vec{\theta}^T \vec{x}_i - y_i \right)}{\partial \vec{\theta}(j)}$$

$$= \sum_{i=1}^{n} 2 \cdot \left(\vec{\theta}^T \vec{x}_i - y_i \right) \vec{x}_i(j)$$

$$\frac{\partial \left(\vec{\theta}^T \vec{x}_i - y_i \right)}{\partial \vec{\theta}(j)} = \frac{\partial (\theta^T \vec{x}_i)}{\partial \vec{\theta}(j)} = \lim_{\epsilon \to 0} \frac{\theta^T \vec{x}_i - (\theta + \epsilon \vec{e}_j)^T \vec{x}_i}{\epsilon} = \lim_{\epsilon \to 0} \frac{\epsilon \vec{e}_j^T \vec{x}_i}{\epsilon} = \vec{x}_i(j).$$

GRADIENT EXAMPLE

Partial derivative for least squares regression:

$$\frac{\partial L(\vec{\theta}, \mathbf{X})}{\partial \vec{\theta}(j)} = \sum_{i=1}^{n} 2 \cdot \left(\vec{\theta}^T \vec{x}_i - y_i \right) \vec{x}_i(j).$$

GRADIENT EXAMPLE

Partial derivative for least squares regression:

$$\frac{\partial L(\vec{\theta}, \mathbf{X})}{\partial \vec{\theta}(j)} = \sum_{i=1}^{n} 2 \cdot \left(\vec{\theta}^T \vec{x}_i - y_i\right) \vec{x}_i(j).$$

$$\vec{\nabla} L(\vec{\theta}, \mathbf{X}) = \sum_{i=1}^{n} 2 \cdot \left(\vec{\theta}^T \vec{x}_i - y_i\right) \vec{x}_i$$

GRADIENT EXAMPLE

Partial derivative for least squares regression:

$$\frac{\partial L(\vec{\theta}, \mathsf{X})}{\partial \vec{\theta}(j)} = \sum_{i=1}^{n} 2 \cdot \left(\vec{\theta}^T \vec{x}_i - y_i\right) \vec{x}_i(j).$$

$$\begin{aligned} \vec{\nabla} L(\vec{\theta}, \mathsf{X}) &= \sum_{i=1}^{n} 2 \cdot \left(\vec{\theta}^T \vec{x}_i - y_i\right) \vec{x}_i \\ &= \mathsf{X}^T(\mathsf{X}\vec{\theta} - \vec{y}). \end{aligned}$$

GRADIENT EXAMPLE

Gradient for least squares regression via linear algebraic approach:

$$\nabla L(\vec{\theta}, \mathsf{X}) = \nabla \|\mathsf{X}\vec{\theta} - \vec{y}\|_2^2$$

GRADIENT DESCENT GREEDY APPROACH

Gradient descent is a greedy iterative optimization algorithm: Starting at $\vec{\theta}^{(0)}$, in each iteration let $\vec{\theta}^{(i)} = \vec{\theta}^{(i-1)} + \eta\vec{v}$, where η is a (small) ‘step size’ and $\vec{v}$ is a direction chosen to minimize $f(\vec{\theta}^{(i-1)} + \eta\vec{v})$.

GRADIENT DESCENT GREEDY APPROACH

Gradient descent is a greedy iterative optimization algorithm: Starting at $\vec{\theta}^{(0)}$, in each iteration let $\vec{\theta}^{(i)} = \vec{\theta}^{(i-1)} + \eta\vec{v}$, where η is a (small) ‘step size’ and $\vec{v}$ is a direction chosen to minimize $f(\vec{\theta}^{(i-1)} + \eta\vec{v})$.

$$D_{\vec{v}} f(\vec{\theta}) = \lim_{\epsilon \to 0} \frac{f(\vec{\theta} + \epsilon\vec{v}) - f(\vec{\theta})}{\epsilon}.$$

GRADIENT DESCENT GREEDY APPROACH

Gradient descent is a greedy iterative optimization algorithm: Starting at $\vec{\theta}^{(0)}$, in each iteration let $\vec{\theta}^{(i)} = \vec{\theta}^{(i-1)} + \eta\vec{v}$, where η is a (small) 'step size' and $\vec{v}$ is a direction chosen to minimize $f(\vec{\theta}^{(i-1)} + \eta\vec{v})$.

$$D_{\vec{v}} f(\vec{\theta}^{(i-1)}) = \lim_{\epsilon \to 0} \frac{f(\vec{\theta}^{(i-1)} + \epsilon\vec{v}) - f(\vec{\theta}^{(i-1)})}{\epsilon}.$$

GRADIENT DESCENT GREEDY APPROACH

Gradient descent is a greedy iterative optimization algorithm: Starting at $\vec{\theta}^{(0)}$, in each iteration let $\vec{\theta}^{(i)} = \vec{\theta}^{(i-1)} + \eta \vec{v}$, where η is a (small) 'step size' and $\vec{v}$ is a direction chosen to minimize $f(\vec{\theta}^{(i-1)} + \eta \vec{v})$.

$$D_{\vec{v}} f(\vec{\theta}^{(i-1)}) = \lim_{\epsilon \to 0} \frac{f(\vec{\theta}^{(i-1)} + \epsilon \vec{v}) - f(\vec{\theta}^{(i-1)})}{\epsilon}.$$

So for small η:

$$f(\vec{\theta}^{(i)}) - f(\vec{\theta}^{(i-1)}) = f(\vec{\theta}^{(i-1)} + \eta \vec{v}) - f(\vec{\theta}^{(i-1)})$$

GRADIENT DESCENT GREEDY APPROACH

Gradient descent is a greedy iterative optimization algorithm: Starting at $\vec{\theta}^{(0)}$, in each iteration let $\vec{\theta}^{(i)} = \vec{\theta}^{(i-1)} + \eta\vec{v}$, where η is a (small) 'step size' and $\vec{v}$ is a direction chosen to minimize $f(\vec{\theta}^{(i-1)} + \eta\vec{v})$.

$$D_{\vec{v}}\, f(\vec{\theta}^{(i-1)}) = \lim_{\epsilon \to 0} \frac{f(\vec{\theta}^{(i-1)} + \epsilon\vec{v}) - f(\vec{\theta}^{(i-1)})}{\epsilon}.$$

So for small η:

$$f(\vec{\theta}^{(i)}) - f(\vec{\theta}^{(i-1)}) = f(\vec{\theta}^{(i-1)} + \eta\vec{v}) - f(\vec{\theta}^{(i-1)}) \approx \eta \cdot D_{\vec{v}} f(\vec{\theta}^{(i-1)})$$

GRADIENT DESCENT GREEDY APPROACH

Gradient descent is a greedy iterative optimization algorithm: Starting at $\vec{\theta}^{(0)}$, in each iteration let $\vec{\theta}^{(i)} = \vec{\theta}^{(i-1)} + \eta\vec{v}$, where η is a (small) 'step size' and $\vec{v}$ is a direction chosen to minimize $f(\vec{\theta}^{(i-1)} + \eta\vec{v})$.

$$D_{\vec{v}}\, f(\vec{\theta}^{(i-1)}) = \lim_{\epsilon \to 0} \frac{f(\vec{\theta}^{(i-1)} + \epsilon\vec{v}) - f(\vec{\theta}^{(i-1)})}{\epsilon}.$$

So for small η:

$$\begin{aligned} f(\vec{\theta}^{(i)}) - f(\vec{\theta}^{(i-1)}) = f(\vec{\theta}^{(i-1)} + \eta\vec{v}) - f(\vec{\theta}^{(i-1)}) &\approx \eta \cdot D_{\vec{v}} f(\vec{\theta}^{(i-1)}) \\ &= \eta \cdot \langle \vec{v}, \vec{\nabla} f(\vec{\theta}^{(i-1)}) \rangle. \end{aligned}$$

GRADIENT DESCENT GREEDY APPROACH

Gradient descent is a greedy iterative optimization algorithm: Starting at $\vec{\theta}^{(0)}$, in each iteration let $\vec{\theta}^{(i)} = \vec{\theta}^{(i-1)} + \eta\vec{v}$, where η is a (small) 'step size' and $\vec{v}$ is a direction chosen to minimize $f(\vec{\theta}^{(i-1)} + \eta\vec{v})$.

$$D_{\vec{v}}\, f(\vec{\theta}^{(i-1)}) = \lim_{\epsilon\to 0} \frac{f(\vec{\theta}^{(i-1)} + \epsilon\vec{v}) - f(\vec{\theta}^{(i-1)})}{\epsilon}.$$

So for small η:

$$\begin{aligned} f(\vec{\theta}^{(i)}) - f(\vec{\theta}^{(i-1)}) = f(\vec{\theta}^{(i-1)} + \eta\vec{v}) - f(\vec{\theta}^{(i-1)}) &\approx \eta \cdot D_{\vec{v}} f(\vec{\theta}^{(i-1)}) \\ &= \eta \cdot \langle \vec{v}, \vec{\nabla} f(\vec{\theta}^{(i-1)}) \rangle. \end{aligned}$$

We want to choose $\vec{v}$ minimizing $\langle \vec{v}, \vec{\nabla} f(\vec{\theta}^{(i-1)}) \rangle$ – i.e., pointing in the direction of $\vec{\nabla} f(\vec{\theta}^{(i-1)})$ but with the opposite sign.

GRADIENT DESCENT PSUEDOCODE

Gradient Descent

- Choose some initialization $\vec{\theta}^{(0)}$.
- For $i = 1, \ldots, t$
  - $\vec{\theta}^{(i)} = \vec{\theta}^{(i-1)} - \eta \nabla f(\vec{\theta}^{(i-1)})$
- Return $\vec{\theta}^{(t)}$, as an approximate minimizer of $f(\vec{\theta})$.

Step size η is chosen ahead of time or adapted during the algorithm (details to come.)

GRADIENT DESCENT PSUEDOCODE

Gradient Descent

- Choose some initialization $\vec{\theta}^{(0)}$.
- For $i = 1, \ldots, t$
  - $\vec{\theta}^{(i)} = \vec{\theta}^{(i-1)} - \eta \nabla f(\vec{\theta}^{(i-1)})$
- Return $\vec{\theta}^{(t)}$, as an approximate minimizer of $f(\vec{\theta})$.

Step size η is chosen ahead of time or adapted during the algorithm (details to come.)

- For now assume η stays the same in each iteration.

GRADIENT DESCENT PSUEDOCODE

Gradient Descent

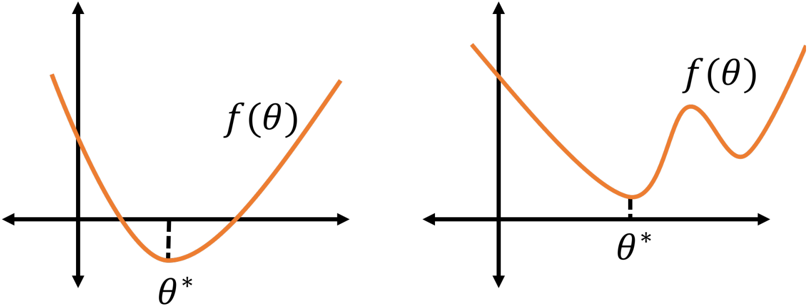
- Choose some initialization $\vec{\theta}^{(0)}$.
- For $i = 1, \ldots, t$
  - $\vec{\theta}^{(i)} = \vec{\theta}^{(i-1)} - \eta \nabla f(\vec{\theta}^{(i-1)})$
- Return $\vec{\theta}^{(t)}$, as an approximate minimizer of $f(\vec{\theta})$.

Step size η is chosen ahead of time or adapted during the algorithm (details to come.)

- For now assume η stays the same in each iteration.

When will this algorithm work well?

$\theta \in \mathbb{R} \quad \nabla f(\theta) \in \mathbb{R}$



Gradient Descent Update: $\vec{\theta}^{(i)} = \vec{\theta}^{(i-1)} - \eta \nabla f(\vec{\theta}^{(i-1)})$

CONDITIONS FOR GRADIENT DESCENT CONVERGENCE

Convex Functions: After sufficient iterations, gradient descent will converge to a approximate minimizer $\hat{\theta}$ with:

$$f(\hat{\theta}) \leq f(\theta_*) + \epsilon$$

CONDITIONS FOR GRADIENT DESCENT CONVERGENCE

Convex Functions: After sufficient iterations, gradient descent will converge to a approximate minimizer $\hat{\theta}$ with:

$$f(\hat{\theta}) \leq f(\theta_*) + \epsilon = \min_{\theta} f(\theta) + \epsilon.$$

CONDITIONS FOR GRADIENT DESCENT CONVERGENCE

Convex Functions: After sufficient iterations, gradient descent will converge to a approximate minimizer $\hat{\theta}$ with:

$$f(\hat{\theta}) \leq f(\theta_*) + \epsilon = \min_{\theta} f(\theta) + \epsilon.$$

Examples: least squares regression, logistic regression, sparse regression (lasso), regularized regression, SVMS,...

CONDITIONS FOR GRADIENT DESCENT CONVERGENCE

Convex Functions: After sufficient iterations, gradient descent will converge to a approximate minimizer $\hat{\theta}$ with:

$$f(\hat{\theta}) \leq f(\theta_*) + \epsilon = \min_{\theta} f(\theta) + \epsilon.$$

Examples: least squares regression, logistic regression, sparse regression (lasso), regularized regression, SVMS,...

Non-Convex Functions: After sufficient iterations, gradient descent will converge to a approximate stationary point $\hat{\theta}$ with:

$$\|\nabla f(\hat{\theta})\|_2 \leq \epsilon.$$

CONDITIONS FOR GRADIENT DESCENT CONVERGENCE

Convex Functions: After sufficient iterations, gradient descent will converge to a approximate minimizer $\hat{\theta}$ with:

$$f(\hat{\theta}) \leq f(\theta_*) + \epsilon = \min_{\theta} f(\theta) + \epsilon.$$

Examples: least squares regression, logistic regression, sparse regression (lasso), regularized regression, SVMS,...

Non-Convex Functions: After sufficient iterations, gradient descent will converge to a approximate stationary point $\hat{\theta}$ with:

$$\|\nabla f(\hat{\theta})\|_2 \leq \epsilon.$$

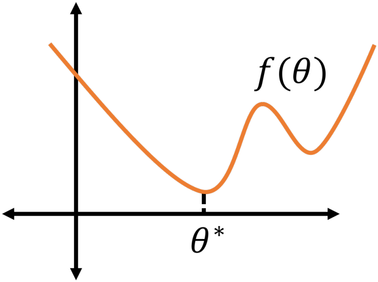
Examples: neural networks, clustering, mixture models.

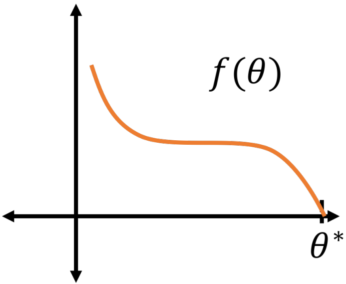
STATIONARY POINT VS. LOCAL MINIMUM

Why for non-convex functions do we only guarantee convergence to a approximate stationary point rather than an approximate local minimum?

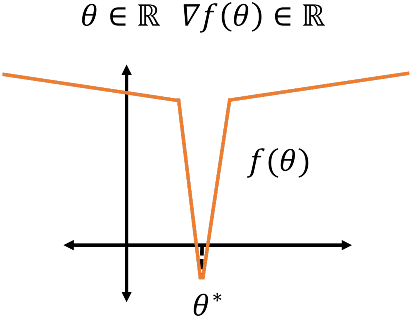
STATIONARY POINT VS. LOCAL MINIMUM

Why for non-convex functions do we only guarantee convergence to a approximate stationary point rather than an approximate local minimum?



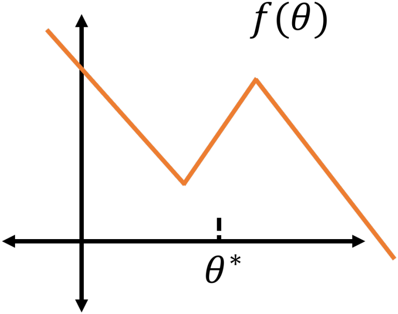


WELL-BEHAVED FUNCTIONS



Gradient Descent Update: $\vec{\theta}^{(i)} = \vec{\theta}^{(i-1)} - \eta \nabla f(\vec{\theta}^{(i-1)})$

WELL-BEHAVED FUNCTIONS



Gradient Descent Update: $\vec{\theta}^{(i)} = \vec{\theta}^{(i-1)} - \eta \nabla f(\vec{\theta}^{(i-1)})$

WELL-BEHAVED FUNCTIONS

Both Convex and Non-convex: Need to assume the function is well behaved in some way.

WELL-BEHAVED FUNCTIONS

Both Convex and Non-convex: Need to assume the function is well behaved in some way.

- Lipschitz (size of gradient is bounded): For all $\vec{\theta}$ and some G,

$$\|\vec{\nabla} f(\vec{\theta})\|_2 \leq G.$$

- Smooth (direction/size of gradient is not changing too quickly): For all $\vec{\theta}_1, \vec{\theta}_2$ and some β,

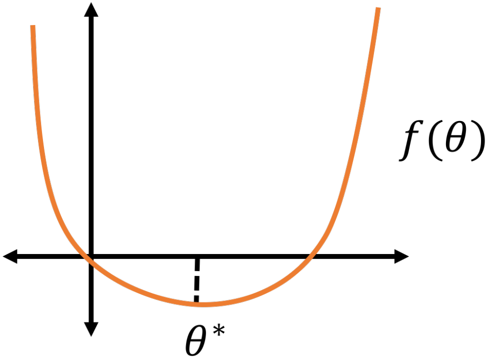
$$\|\vec{\nabla} f(\vec{\theta}_1) - \vec{\nabla} f(\vec{\theta}_2)\|_2 \leq \beta \cdot \|\vec{\theta}_1 - \vec{\theta}_2\|_2.$$

Gradient Descent analysis for convex functions.

CONVEXITY

Definition – Convex Function: A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if and only if, for any $\vec{\theta}_1, \vec{\theta}_2 \in \mathbb{R}^d$ and $\lambda \in [0, 1]$:

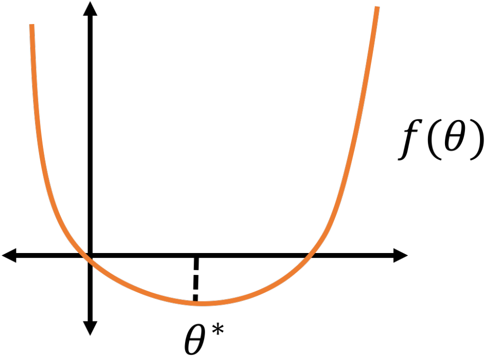
$$(1-\lambda)\cdot f(\vec{\theta}_1) + \lambda \cdot f(\vec{\theta}_2) \geq f\left((1-\lambda)\cdot \vec{\theta}_1 + \lambda \cdot \vec{\theta}_2\right)$$



CONVEXITY

Corollary – Convex Function: A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if and only if, for any $\vec{\theta}_1, \vec{\theta}_2 \in \mathbb{R}^d$ and $\lambda \in [0, 1]$:

$$f(\vec{\theta}_2) - f(\vec{\theta}_1) \geq \vec{\nabla} f(\vec{\theta}_1)^T \left(\vec{\theta}_2 - \vec{\theta}_1\right)$$



GD ANALYSIS – CONVEX FUNCTIONS

Assume that:

- f is convex.
- f is G Lipschitz (i.e., $\|\vec{\nabla} f(\vec{\theta})\|_2 \leq G$ for all $\vec{\theta}$.
- $\|\vec{\theta}_0 - \vec{\theta}_*\|_2 \leq R$ where θ_0 is the initialization point.

Gradient Descent

- Choose some initialization $\vec{\theta}_0$ and set $\eta = \frac{R}{G\sqrt{t}}$.
- For $i = 1, \ldots, t$
  - $\vec{\theta}_i = \vec{\theta}_{i-1} - \eta \nabla f(\vec{\theta}_{i-1})$
- Return $\hat{\theta} = \arg\min_{\vec{\theta}_0, \ldots \vec{\theta}_t} f(\vec{\theta}_i)$.

GD ANALYSIS PROOF

Theorem – GD on Convex Lipschitz Functions: For convex G Lipschitz function f, GD run with $t \geq \frac{R^2G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G\sqrt{t}}$, and starting point within radius R of θ_*, outputs $\hat{\theta}$ satisfying:

$$f(\hat{\theta}) \leq f(\theta_*) + \epsilon.$$

GD ANALYSIS PROOF

Theorem – GD on Convex Lipschitz Functions: For convex G Lipschitz function f, GD run with $t \geq \frac{R^2G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G\sqrt{t}}$, and starting point within radius R of θ_*, outputs $\hat{\theta}$ satisfying:

$$f(\hat{\theta}) \leq f(\theta_*) + \epsilon.$$

Step 1: For all i, $f(\theta_i) - f(\theta_*) \leq \frac{\|\theta_i - \theta_*\|_2^2 - \|\theta_{i+1} - \theta_*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$. Visually:

GD ANALYSIS PROOF

Theorem – GD on Convex Lipschitz Functions: For convex G Lipschitz function f, GD run with $t \geq \frac{R^2G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G\sqrt{t}}$, and starting point within radius R of θ_*, outputs $\hat{\theta}$ satisfying:

$$f(\hat{\theta}) \leq f(\theta_*) + \epsilon.$$

Step 1: For all i, $f(\theta_i) - f(\theta_*) \leq \frac{\|\theta_i - \theta_*\|_2^2 - \|\theta_{i+1} - \theta_*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$. Formally:

GD ANALYSIS PROOF

Theorem – GD on Convex Lipschitz Functions: For convex G Lipschitz function f, GD run with $t \geq \frac{R^2G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G\sqrt{t}}$, and starting point within radius R of θ_*, outputs $\hat{\theta}$ satisfying:

$$f(\hat{\theta}) \leq f(\theta_*) + \epsilon.$$

Step 1: For all i, $f(\theta_i) - f(\theta_*) \leq \frac{\|\theta_i - \theta_*\|_2^2 - \|\theta_{i+1} - \theta_*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$.

Step 1.1: $\nabla f(\theta_i)(\theta_i - \theta_*) \leq \frac{\|\theta_i - \theta_*\|_2^2 - \|\theta_{i+1} - \theta_*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$

GD ANALYSIS PROOF

Theorem – GD on Convex Lipschitz Functions: For convex G Lipschitz function f, GD run with $t \geq \frac{R^2G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G\sqrt{t}}$, and starting point within radius R of θ_*, outputs $\hat{\theta}$ satisfying:

$$f(\hat{\theta}) \leq f(\theta_*) + \epsilon.$$

Step 1: For all i, $f(\theta_i) - f(\theta_*) \leq \frac{\|\theta_i - \theta_*\|_2^2 - \|\theta_{i+1} - \theta_*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$.

Step 1.1: $\nabla f(\theta_i)(\theta_i - \theta_*) \leq \frac{\|\theta_i - \theta_*\|_2^2 - \|\theta_{i+1} - \theta_*\|_2^2}{2\eta} + \frac{\eta G^2}{2} \implies$ Step 1.

GD ANALYSIS PROOF

Theorem – GD on Convex Lipschitz Functions: For convex G Lipschitz function f, GD run with $t \geq \frac{R^2G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G\sqrt{t}}$, and starting point within radius R of θ_*, outputs $\hat{\theta}$ satisfying:

$$f(\hat{\theta}) \leq f(\theta_*) + \epsilon.$$

Step 1: For all i, $f(\theta_i) - f(\theta_*) \leq \frac{\|\theta_i - \theta_*\|_2^2 - \|\theta_{i+1} - \theta_*\|_2^2}{2\eta} + \frac{\eta G^2}{2}$

GD ANALYSIS PROOF

Theorem – GD on Convex Lipschitz Functions: For convex G Lipschitz function f, GD run with $t \geq \frac{R^2G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G\sqrt{t}}$, and starting point within radius R of θ_*, outputs $\hat{\theta}$ satisfying:

$$f(\hat{\theta}) \leq f(\theta_*) + \epsilon.$$

Step 1: For all i, $f(\theta_i) - f(\theta_*) \leq \frac{\|\theta_i - \theta_*\|_2^2 - \|\theta_{i+1} - \theta_*\|_2^2}{2\eta} + \frac{\eta G^2}{2} \implies$

Step 2: $\frac{1}{T}\sum_{i=1}^{T} f(\theta_i) - f(\theta_*) \leq \frac{R^2}{2\eta \cdot T} + \frac{\eta G^2}{2}$.

GD ANALYSIS PROOF

Theorem – GD on Convex Lipschitz Functions: For convex G Lipschitz function f, GD run with $t \geq \frac{R^2G^2}{\epsilon^2}$ iterations, $\eta = \frac{R}{G\sqrt{t}}$, and starting point within radius R of θ_*, outputs $\hat{\theta}$ satisfying:

$$f(\hat{\theta}) \leq f(\theta_*) + \epsilon.$$

Step 2: $\frac{1}{T}\sum_{i=1}^{T} f(\theta_i) - f(\theta_*) \leq \frac{R^2}{2\eta \cdot T} + \frac{\eta G^2}{2}$.

Questions on Gradient Descent?